الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

présenté par

Saddok Omar

pour l'obtention du diplôme master2 en Électronique option Systèmes de Vision et Robotique

Thème

Détection des zones navigables par contours actifs: Application au Robucar

Proposé par : Mme AZOUAOUI OUAHIBA & Mr. DJEKOUNE A.OUALID

Mme BOUGHERIRA NADIA

Année Universitaire 2011-2012

Remerciements

Tout d'abord, je souhaite manifester mes sincères remerciements à ***ALLAH*** *le tout puissant qui m'a donné la force, la patience, le courage, la volonté et la santé durant toutes ces années d'étude pour aller jusqu'au bout de ce travail.*

C'est avec grand plaisir que je réserve cette page, en signe de gratitude à tous ceux qui m'ont aidé à la réalisation de mon projet de fin d'étude.

J'exprime toute ma gratitude pour mes Promoteurs M^{elle} ***Azouaoui Ouahiba*** *et* ***M^{r} Djekoune A. Oualid*** *du CDTA pour leurs précieux conseils, leur disponibilité, leur gentillesse et simplicité, la confiance qu'ils m'ont toujours témoignée et la sollicitude dont ils m'ont entouré. Ainsi que Mr* ***A.Ellarbi*** *pour leur présence et leur aide.*

Je remercie ma Co-promotrice ***M$^{m\ e}$ N.BOUGHERIRA*** *pour ses conseils et orientations tant précieuses qu'elle m'a prodiguée durant ce projet.*

Je tiens à remercier l'ensemble des chercheurs, travailleurs du CDTA et en particulier l'ensemble de l'équipe NCRM de la division productique et robotique pour leurs aides et leurs conseils ; ainsi que tous les enseignants de département d'électronique et surtout du master Systèmes de Vision et Robotique de l'université de Blida pour leurs disponibilités et leurs directives.

Je remercie aussi chaleureusement les membres du jury, pour l'honneur qu'ils m'ont fait en acceptant d'évaluer mon travail.

ET enfin, un grand remerciement à mes chères parents, mes sœurs, mes frères et toute ma famille pour leur soutien indéfectible sans oublié tous mes camarades, mes amis et tous ceux qui m'ont connu de proche ou de loin.

Dédicaces

Je dédie ce modeste travail
A mes très chers parents qui n'ont jamais cessé de m'encourager, que dieu les protège et les offre une langue vie pleine de paix et de joie

A mes très chers frères Djilali, Ilias et à ses enfants et ses femmes
A mes très chères frères Missoum, Farouk et Soufien
A mes très chères sœurs Fatouma, Saliha et Fatiha et à ses enfants....
A mes très chères sœurs Saida, Salima

A toute ma grande famille....

A tous mes cousins et mes voisins.....

A mes amis Abdenour, Bilal, Youcef,...

A la mémoire de mon défunt cher ami Rabah

A tous mes ami(e)s avec lesquels j'ai passé d'agréables moments
Et à tous ceux qui m'aiment et que j'aime

Avec l'expression de tous mes sentiments de respect,

Je dédie ce modeste travail.

Omar...

Résumé

ملخص: العمل المقدم في هذه المذكرة يتعلق بالكشف والتتبع البصري لمناطق الملاحة، في وقت آنى، من أجل الملاحة الخارجية لروبوت متحرك ذات تحكم ذاتي من نوع سيارة باستخدام كاميرا محمولة. من الصورة الحالية، الروبوت المتحرك يحدد ويحد المنطقة حيث يمكنه التنقل متجنبا الاصطدام مع العقبات في محيطه (المحيط الخارجي).

لهذ الغرض، يتم استخدام طريقة لتقطيع الصور باستخدام الحواف النشطة (الثعابين) للكشف عن هذه المناطق. لأنها تقوم على أساس التقليل من الطاقة الوظيفية بواسطة خوارزمية قريدي.

كلمات المفاتيح: التتبع البصري، مناطق الملاحة، روبوت متحرك ذات تحكم ذاتي نوع سيارة، تقطيع الصور، الحواف النشطة، الطاقة الوظيفية، خوارزمية قريدي.

Résumé : Le travail présenté dans ce mémoire concerne la détection et le suivi visuel des zones navigables, en temps réel, pour la navigation d'extérieur d'un robot mobile autonome de type voiture à l'aide d'une caméra embarquée. A partir de l'image courante, le robot mobile détermine et limite la zone où il peut naviguer sans contrainte de collision avec les obstacles de son environnement (environnement d'extérieur structuré).

A cet effet, une méthode de segmentation d'images utilisant les contours actifs (Snakes) est utilisée pour la détection de ces zones. Elle est basée sur une minimisation d'une fonctionnelle d'énergie par l'algorithme de Greedy.

Mots clefs : Suivi visuel, zones navigables, robot mobile autonome de type voiture, segmentation d'images, contours actifs, fonctionnelle d'énergie, algorithme de Greedy.

Abstract: The work presented in this memory concerns the detection and visual tracking, in real time, of navigable zones in outdoor navigation of a car like autonomous mobile robot with an embarked camera. From the current image, the mobile robot determines and limits the area where it can navigate without constraint of collision with obstacles in its environment (external structured environment). However, an image segmentation method using active contours (snakes) is used for the detection of these areas. It is based on a minimization of a functional energy by the Greedy algorithm.

Résumé

Keywords: Visual tracking, navigables zones, autonomous mobile robot type car, image segmentation, active contours, functional energy, Greedy algorithm.

Liste des Acronymes et Abréviations

Nous donnons ici quelques acronymes et quelques définitions qui interviennent fréquemment dans ce mémoire. Nous avons essayé de les définir dans la plupart des cas lors de leur première utilisation, sauf pour les notions les plus courantes.

Amers : les amers sont des objets qui du fait de leur saillance par rapport à l'environnement, servent un robot mobile pour se repérer lorsqu'il se déplace.

Cyber-car : c'est un robot mobile autonome de type voiture de troisième génération, il est capable de transporter des passagers et de circuler dans les réseaux routiers.

CDTA : Centre de Développement des Technologies Avancées.

DPI : Dots Per Inch.

DILIGENT: un robot mobile d'intérieur Nomadic XR4000 de LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes), Toulouse, France.

GGVF : Generalized Gradient Vector Flow (gradient vecteur flou généralisé).

GVF : Gradient Vector Flow (gradient vecteur flou).

INRIA : Institut National de la Recherche en Informatique et Automatique (France).

Intensité Lumineuse : c'est l'énergie lumineuse émise par une source dans un angle solide.

ITS : Intensité, Teinte & Saturation.

Luminosité : c'est la perception visuelle de la luminance. La luminosité fait référence à la quantité de lumière qui est absorbée ou réfléchie par une zone colorée.

LAMA : le robot Lama est un châssis à six roues de conception russe, acquis par Alcatel en 1995, prêté au LAAS depuis 1996.

NCRM : équipe Navigation et Contrôle des Robots Mobiles Autonomes du CDTA.

NDG : Niveau de gris.

PPP : Point Par Pixel.

ROI: Region of Interest.

RVB: Rouge, Vert & Blue.

SLAM: Simultaneous Localization and Mapping (Localisation et Cartographie Simultanées).

Table des matières

Table des matières

Table des matières

Table des matières

Liste des figures

Liste des figures

Liste des tableaux

Introduction Générale

Introduction Générale

De nos jours les robots mobiles autonomes intelligents, qui réalisent des tâches sans intervention d'un opérateur, sont nécessaires dans plusieurs domaines d'application et peuvent ainsi accomplir des tâches complexes dans divers environnements par eux-mêmes tel un être humain. Pour cela, ces robots sont équipés de capacités de perception, de décision et d'action qui leur permettent d'agir de manière autonome dans leur environnement en fonction de la perception qu'ils en ont[1].Les robots mobiles sont largement utilisés dans divers environnements : industrie, transport, service, médicale, etc.

Pour cela là, la navigation des robots mobiles est devenue un champ de recherche très vaste pour la communauté de la vision et la robotique. Son problème consiste à intégrer dans un robot réel, toutes les fonctions nécessaires pour qu'il puisse exécuter des déplacements à travers un environnement connu ou inconnu; en utilisant les données perçues par ses capteurs. Pour obtenir cette navigation, il faut avoir au moins une méthode de suivi visuel intégrée sur le robot.

1 Travaux antérieurs :

Plusieurs travaux ont été réalisé par l'équipe Navigation et Contrôle des Robots Mobiles Autonomes (NCRM) du Centre de Développement des Technologies Avancées (CDTA) proposant des solutions à la navigation d'un robot mobile de type voiture Robucar se déplaçant dans un milieu d'extérieur structuré dont le but de réaliser des voitures autonomes intelligentes de troisième génération connues sous le nom de Cyber-Cars[2]. Ces dernières, sont capables de naviguer dans des environnements urbains sans intervention humaine, et destinées pour le transport automatisé.

Parmi ces travaux, nous citons :

Introduction Générale

1.1 Détection des bords de la chaussée par vision monoculaire pour robot mobile de type voiture : application au transport urbain [3].

Dans ce travail, une approche de détection et suivi des bords de la chaussée a été développée, où la contribution consistait à introduire des fonctionnalités visuelles dans le système embarqué. Cette approche utilise les informations visuelles issues de caméra embarquée ; les images acquises sont prétraitées pour améliorer leurs qualités, puis sélectionne la bonne droite, qui limite la chaussée et le trottoir, à suivre en utilisant la transformée de Hough, permettant au Robucar de naviguer d'une manière autonome dans cet environnement.

1.2 Contribution au développement d'un système de détection des bords de la chaussée par vision monoculaire pour un robot mobile autonome [4].

Dans le même contexte que précédemment, l'approche développée utilise l'information image pour détecter le bord de la chaussée, et un capteur laser pour détecter les obstacles. L'image acquise doit être prétraitée dans le but d'obtenir de meilleurs résultats. En effet, ceci est effectué grâce à l'utilisation automatique des deux prétraitements à savoir l'égalisation et l'élargissement de l'histogramme des images acquises. Une détection de contours est réalisée par le filtre de Deriche, quant à la détection de la droite représentant le bord de la chaussée, elle est réalisée par la transformée de Hough. Une fois la droite choisie, une stratégie de suivi est mise en œuvre. Cette stratégie permet au robot mobile Robucar de naviguer d'une manière autonome dans un environnement de type route.

La partie de détection d'obstacles est faite grâce à un capteur laser fixé sur le robot. La stratégie de détection et d'évitement a été développée de manière à interrompre le suivi de bord de la chaussée tout en effectuant l'évitement, dans le cas où un éventuel obstacle est détecté sur la trajectoire du robot.

Pour améliorer ces travaux, et donner un plus à l'application demandée, nous étudions à notre tour la détection visuelle des zones navigables de la chaussée pour le Robucar en temps réel, en utilisant une méthode de segmentation d'images basée sur les Snakes.

2 Contribution :

Ce travail est réalisé dans le cadre de projet de fin d'étude pour l'obtention du diplôme master2 en électronique ; dont l'objectif est de permettre à un robot mobile de type voiture, Robucar de détecter et de suivre les zones navigables de la chaussée se trouvant sur son chemin, et ceci, en utilisant la vision monoculaire. Sachant que ce robot se déplace dans un milieu urbain structuré, avec présence de plusieurs objets mobiles et fixes tels que d'autres véhicules, piétons, etc. Notre mission est de faire de telles sortes que ce robot mobile arrive à naviguer, sans collisions, tout en connaissant les zones navigables contenues parmi les obstacles de son environnement.

Afin de réaliser la détection des zones navigables de la chaussée, nous utilisons pour notre cas la **vision monoculaire**. Il est évident que le système complet de navigation, pourra exploiter d'autres capteurs pour avoir une bonne perception du monde dans lequel le robot doit évoluer, y compris des capteurs proprioceptifs (odométrie, gyroscope, etc.) ou des capteurs extéroceptifs (télémètre laser, infrarouge, etc.).

Le travail présenté dans ce mémoire concernent le thème de détection et suivi visuel des zones navigables en environnement d'extérieur de type route appliqué au robot mobile Robucar, en utilisant une méthode de segmentation d'image basée sur lescontours actifs (snakes) introduite par les auteurs Kass,Witkin et Terzopoulousdans [5], et qui se base sur le principe de minimisation d'une fonctionnelle d'énergie pour faire épouser une forme ou un objet ; mais nous nous intéressons à la détection ensuite le suivi des zones navigables en temps réels.

Introduction Générale

Pour commencer dans le chapitre 1, nous allons circonscrire nos travaux d'une manière générale, à l'intérieur de la problématique de la navigation autonome d'un robot mobile, par la suite aborder la problématique spécifique de la navigation dans deux environnements, qui seront traités dans ce mémoire. D'abord, nous commencerons par définir la robotique mobile autonome, et citer quelques notions sur la perception et les différents capteurs utilisés. Ensuite, nous parlerons de la navigation en milieu d'intérieur, avec deux approches différentes. Par la suite, nous évoquerons la navigation d'un robot mobile en environnement d'extérieur, où nous distinguerons aussi deux approches. Et enfin, nous décrivons notre méthode de détermination des zones navigables pour le robot mobile « Robucar » en milieu d'extérieur structuré.

Dans le chapitre 2, nous allons décrire de manière générale un état de l'art, où nous présentons en premier lieu quelques notions fondamentales utilisées dans le traitement d'images, ainsi que les techniques de prétraitement et de segmentation existantes dans la littérature. Nous nous intéresserons par la suite aux méthodes de segmentation par contours actifs classiques (snakes). Ensuite nous introduisons les opérateurs des algorithmes de Greedy à cette méthode afin de palier à ses inconvénients.

Enfin, dans le dernier chapitre nous présenterons notre approche développée où nous donnons une description de notre application;ainsi, quelques illustrations et résultats des tests effectués.

Finalement nous terminons par une conclusion générale et nous donnons quelques perspectives concernant ce travail.

Chapitre 1 : Navigation Des Robots Mobiles Autonomes

1.1 Introduction :

La robotique mobile est un champ de recherche très varié, du fait principalement des nombreuses applications potentielles. Afin de réaliser des tâches dans le cadre d'une de ces applications, un robot doit posséder la capacité de naviguer dans son environnement. La navigation d'un robot mobile est devenue un des problèmes clés dans la communauté de la robotique. Grossièrement, le problème consiste à intégrer dans un robot réel, toutes les fonctions nécessaires pour qu'il puisse exécuter des déplacements à travers un environnement donné, en utilisant l'information perçue par des capteurs. Il existe plusieurs approches pour résoudre cette tâche, mais entre toutes ces approches, les plus connues et aussi les plus utilisées sont fondées sur la construction d'un modèle du Monde à partir de l'information sensorielle [6] ; le robot utilisera ce modèle pour exécuter des mouvements requis par une tâche de plus haut niveau.

Par exemple, la navigation en milieu intérieur, le robot pourra recevoir un ordre du type « *Aller dans la pièce C* ». Le modèle de l'environnement sera exploité que cela soit afin de se localiser (« *Je suis dans la pièce A* »), pour choisir un chemin dans une description topologique (« *Pour aller dans la pièce C depuis la pièce A, je dois traverser la pièce B, puis le couloir T* »), pour générer une trajectoire (« *Pour traverser la pièce B, je dois suivre telle courbe sur le sol, définie dans un repère RB* », « *Pour traverser le couloir T, suivre les mur, etc.* ») ou pour contrôler l'exécution des déplacements planifiés (« *Pour passer de la pièce B au couloir T, chercher la porte, la franchir, etc.* »). Le problème de la navigation peut donc être décomposé en plusieurs tâches, comme par exemple : la planification de trajectoires, l'évitement d'obstacles, etc. C'est la partie navigation qui nous intéresse dans le cadre de ce mémoire, et plus précisément la navigation du robot mobile Robucar dans un environnement d'extérieur structuré de type route.

Dans notre cas, la vision est choisie comme capteur privilégié du fait de la richesse de l'information perçue : résolution, information aussi bien géométrique que

radiométrique (luminance, couleur), etc. Nos travaux s'appliquent aussi bien à la navigation dans des environnements d'extérieur (suivi des bords d'une route, suivi d'un autre véhicule) ; que d'intérieur (suivi des bords d'un couloir, d'une porte, suivi d'amers de type affiche, etc.).

Dans ce chapitre, nous introduisons le contexte de notre travail concernant la détection et le suivi visuel des zones navigables embarqué dans un robot mobile. Tout d'abord, nous commençons par définir la robotique mobile autonome ; et citer quelques notion sur elle; tels que l'autonomie, l'holonomie, non holonomie. Aussi, nous parlons d'un point clé en robotique mobile qui est la perception de l'environnement. Dans cette optique, nous allons étudier les différents outils permettant aux robots de percevoir son environnement et de s'y repérer. Parmi celles-ci, nous citons : les capteurs proprioceptifs (odomètres, gyromètres, etc.) et les capteurs extéroceptifs (capteurs tactiles, télémètres, etc.).

Par la suite, nous s'intéressons à la navigation des robots mobiles; où nous précisons les différents étapes et approches de navigation, son oublié les aspects spécifiques aux deux principaux contextes applicatifs : la navigation en milieu intérieur (robots de service, robots domestiques…) et la navigation en milieu extérieur (véhicules intelligents,…).

Enfin, pour terminer, nous décrivons notre approche de navigation qui consiste à détecter et suivi un environnement d'extérieur structuré de type route ; où nous nous basons sur la détection visuelle des régions navigables. Cette détection est faite à l'aide d'une caméra embarquée sur le Robucar, en utilisant une méthode de segmentation par contours actifs.

1.2 Robotique mobile autonome :

1.2.1 Définition d'un robot mobile autonome :

Les Robots Mobiles Autonomes (RMA) sont nés avec la robotique dite de troisième génération et sont donc des machines capables avant tout de réaliser le lien entre la perception et l'action. Ils sont capables de naviguer de manière autonome dans un environnement complexe, parfois évolutif, partiellement connu ou inconnu, et d'exécuter des tâches programmées sans intervention humaine ou avec une intervention réduite [7].

Alors, un RMA est une machine physique qui agit sur son environnement pour atteindre un objectif.

La figure 1.1 illustre quelques exemples de RMA.

(a)

(b)

(c)

Figure 1.1 : Exemples de robots mobiles autonomes.

a) Robot d'extérieur planétaire b) Robot d'intérieur c) robot urbain (cyber-car)

1.2.1.1 Autonomie :

Un système est dit autonome s'il est capable d'adapter son comportement à l'environnement [8]. On dit qu'un robot mobile est autonome s'il vérifie ces deux conditions : la versatilité et l'autoadaptativité.

- **Versatilité :**

La versatilité signifie que le robot devrait être capable d'effectuer des tâches diverses de plusieurs manières différentes [9].

- **Auto adaptativité :**

L'auto adaptativité veut dire que le robot devrait être capable d'accomplir correctement sa tâche ; même s'il rencontre de nouvelles situations inattendues, sans intervention humaine [9].

1.2.1.2 Notion d'holonomie :

La notion d'holonomie signifie la capacité d'un robot mobile à pouvoir se déplacer à partir d'une situation dans n'importe quelle direction. En effet, le mécanisme holonome permet au robot de manœuvrer dans n'importe quelle direction arbitraire à partir de n'importe quelle configuration arbitraire. Il n'y a pas de contrainte sur les commandes. Cela signifie une très grande simplification des problèmes de contrôle [10].

En effet, un système holonome est un système qui a le même nombre de déplacement virtuels que le nombre de coordonnées généralisées pour le décrire. La figure 1.2 illustre un robot holonome omnidirectionnel qui peut faire trois mouvements planaires indépendants ; deux translations l'un avancer ou reculer, l'autre aller vers la droite ou vers la gauche ; et une rotation : tourner vers la droite ou vers la gauche. Ces mouvements sont admissibles à une vitesse non nulle à partir d'une configuration quelconque [11].

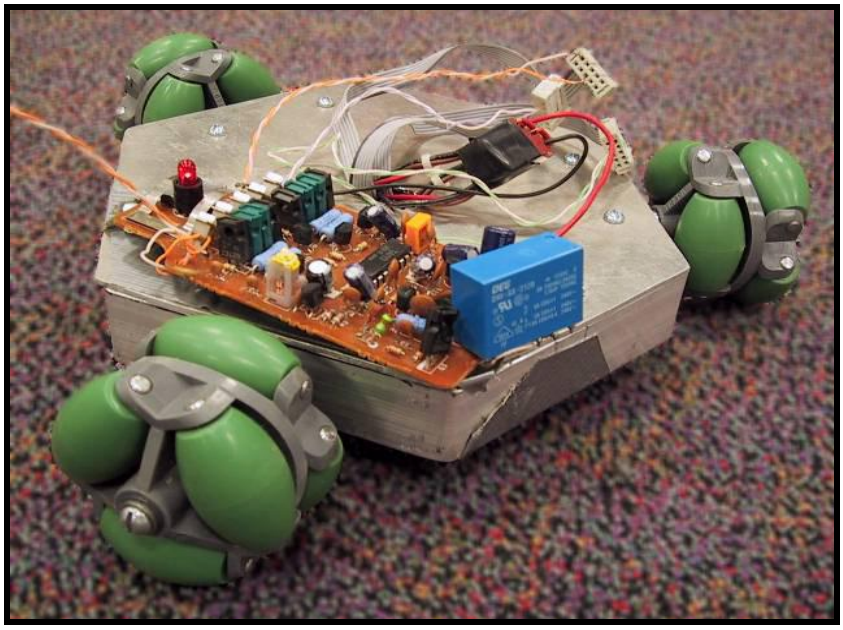


Figure 1.2: Exemple d'un robot mobile holonome.

1.2.1.3 Notion de non holonomie :

Le mouvement d'un système non holonome est produit seulement par deux déplacements indépendants, donc le robot ne possède que deux degrés de liberté. Les systèmes possédant des contraintes non holonomes nécessitent toujours un nombre plus grand de coordonnées pour leur description que le nombre de degrés de liberté [12]. La contrainte non holonome est la limitation des vitesses admissibles d'un objet, ce qui veut dire que ce robot ne peut effectuer de mouvement instantanément que dans certaines directions [12].

1.2.2 Perception :

La notion de perception en robotique mobile est relative à la capacité du robot mobile à recueillir, traiter et mettre en forme des informations qui lui sont utiles pour agir et réagir dans le monde qui l'entoure. Alors que pour des tâches de manipulation on peut considérer que l'environnement du robot est relativement structuré, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement connus [13]. Aussi, pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée. Pour se focaliser sur le problème de navigation, nous nous restreignons dans ce qui suit aux capteurs utiles à cette tâche.

1.2.2.1 Capteurs :

1.2.2.1.1 Définition :

Les capteurs sont des dispositifs permettant de transformer une grandeur physique (la valeur mesurée), la plus souvent environnementale en une autre grandeur physique (*électrique*) interprétable par un système. Les capteurs qui équipent un robot mobile doivent assurer la perception de l'environnement, et la localisation dans ce dernier.

1.2.2.1.2 Différents capteurs utilisés en robotique :

Les capteurs utilisés en robotique sont classés en deux catégories ; des capteurs *extéroceptifs* qui informent le robot sur la nature de l'environnement, et des capteurs *proprioceptifs* qui mesurent l'état interne du robot [4].

1.2.2.1.2.1 Capteurs proprioceptifs:

Les capteurs proprioceptifs sont installés à bord de robot ; ils fournissent des informations propres au comportement interne du robot, c'est à-dire sur son état à un instant donné, tels que la vitesse des roues, l'angle des articulateurs, etc. [4]. Nous regroupons les capteurs proprioceptifs en deux familles : les capteurs de déplacement (odomètre, accéléromètre, radar doppler, etc.) et les capteurs d'attitude (gyroscope, gyromètre, etc.)

1.2.2.1.2.2 Capteurs extéroceptifs:

Ces capteurs informent le robot sur son environnement. L'œil est une parfaite illustration de capteur extéroceptif chez l'homme. Dans un robot, de tels capteurs sont essentiellement les caméras, qui lui permettent de percevoir l'aspect visuel des objets et éventuellement de calculer leur structure 3D, leur position, leur vitesse. D'autres types de capteurs, à savoir, les télémètres (sonars, radar, lasers) sont utilisés et fournissent directement des informations sur la géométrie du monde qui entoure le robot [4] [14].

1.2.2.2 Caméras :

L'utilisation d'une caméra pour percevoir l'environnement est une méthode attractive car elle semble proche des méthodes utilisées par les êtres humains. Le traitement des données volumineuses et complexes fournies par ces capteurs reste cependant difficile à l'heure actuelle, même si cela reste une voie de recherche très explorée [15].

Les caméras actuelles sont constituées par des boîtiers compacts sur lesquels se vissent les objectifs et qui contiennent un capteur photosensible logé derrière un verre protecteur. Le rôle du capteur est de transformer la quantité de lumière incidente en une charge électrique par effet photo-électrique, et de convertir cette charge en une grandeur électrique mesurable. Les capteurs sont décomposés en petites cellules jointives indépendantes, qui reçoivent chacune une petite partie de l'image, c'est-à-dire un pixel [4].

1.3 Navigations des robots mobiles autonomes :

1.3.1 Introduction :

La navigation d'un robot mobile est un des problèmes clés dans la robotique. Alors, le problème consiste à intégrer dans un robot, toutes les fonctions nécessaires pour qu'il puisse exécuter des déplacements dans un environnement connu, en se basant sur l'information perçue par ses capteurs [6]. Il existe plusieurs approches pour résoudre ce problème, mais parmi celles-ci, nous citons celle qui est fondée sur la construction d'un modèle du monde à partir des informations issues par les différents capteurs. Le robot utilisera ce modèle pour exécuter des mouvements requis par une tâche de plus haut niveau.

Une tâche essentielle à l'autonomie des robots est de naviguer en toute sécurité dans un environnement inconnu. L'apprentissage d'un lieu inconnu peut se faire par l'intermédiaire d'une phase de télé opération du robot. Les données issues d'un capteur d'image embarqué sur le robot sont analysées pour extraire les informations caractéristiques de l'image. Ces informations sont utilisées pour construire des modèles représentatifs des amers naturels trouvés dans la scène observée, dans le but de faire naviguer le robot.

La navigation est définie comme le procédé permettant de répondre aux trois questions suivantes [13] :

1-≪où suis-je? ≫ : se localiser dans son environnement.

2-≪où dois-je aller? ≫ : trouvé des zones d'intérêt à explorer ou des objets dans son environnement et lies à sa mission.

3-≪comment dois-je y aller? ≫ : de planifier ses actions, pour par exemple définir une trajectoire pour se rendre d'un point A à un point B.

1.3.2 Etapes de navigation :

Il existe trois étapes dans la navigation, et qu'ils sont la perception et modélisation, décision et planification, et enfin l'exécution. La figure 1.3 illustre une chaîne fonctionnelle d'un système de navigation.

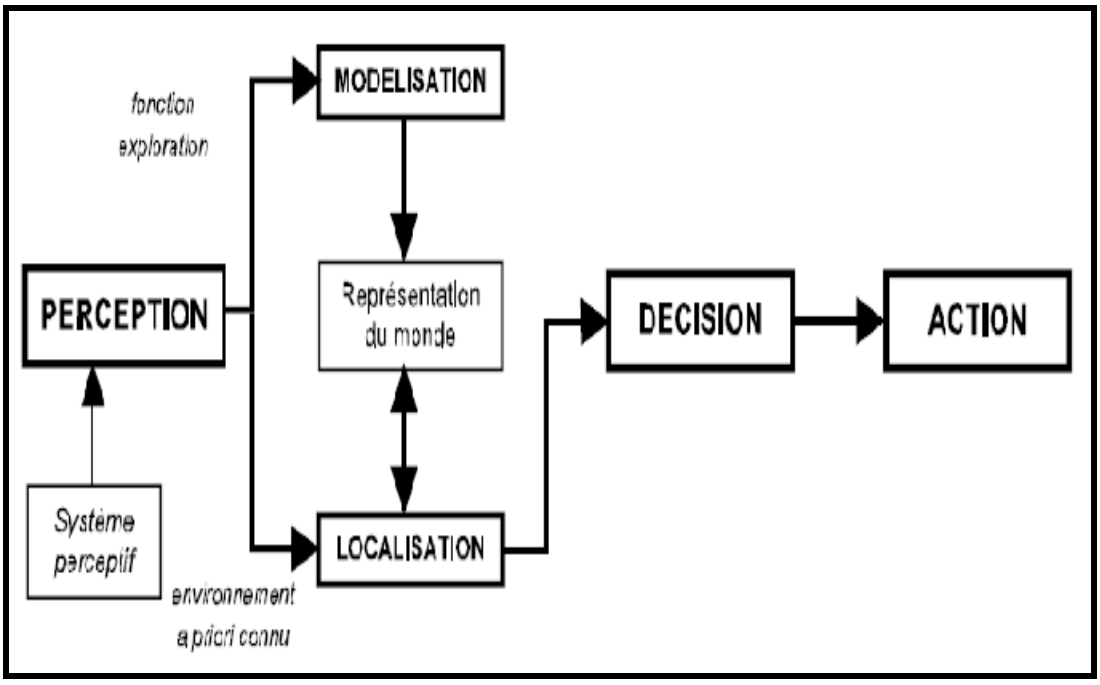


Figure 1.3 : Chaîne fonctionnelle d'un système de navigation.

1.3.2.1 Perception et Modélisation :

La perception permet de voir l'environnement proche et éloigné du robot. Elle est nécessaire pour la sécurité du robot, pour la modélisation de l'environnement et pour sa localisation dans ce dernier. [4].

L'étape de modélisation a pour but de maintenir un modèle d'environnement. Ce modèle intègre les données issues de l'environnement [16]. Afin de naviguer dans celui-ci, le robot doit soit construire son propre modèle du monde, soit l'acquérir. En plus, il existe plusieurs possibilités pour les représentations utilisées dans ce modèle du monde, parmi lesquelles on cite [4] :

- Une représentation géométrique, (par exemple un ensemble de segments laser).

- Une représentation topologique, qui décrit les connexions entre couloirs et pièces.

1.3.2.2 Décision et Planification :

Les informations en provenance des différents capteurs doivent être interprétées comme autant d'éléments utiles à la prise de décision sur l'action à faire, le but étant de délivrer les ordres corrects aux actionneurs, les moteurs des roues, etc. Avec un robot mobile, il est nécessaire de donner des priorités en fonction des informations reçues. Par exemple, si un capteur de contact informe d'un choc sur l'avant, cette information est prioritaire sur un déplacement du robot vers l'avant et doit entraîner un arrêt ou un déplacement vers une autre direction [17].

1.3.2.3 Exécution :

Le robot dispose d'actionneurs ≪ moteurs ≫ permettant son déplacement. Le rôle de ces actionneurs est d'exécuter les commandes correspondant aux décisions.

1.3.3 Approches de navigation :

Il existe plusieurs méthodes de navigation qui sont regroupées en trois approches :

-Approche globale.

-Approche locale.

-Approche mixte ou hybride.

1.3.3.1 Approche globale :

Ce type d'approches est propre à un univers parfaitement connu. On extrait un modèle à travers lequel l'environnement est le plus fidèlement représenté [18]. Ces approches consistent à capturer la complexité de l'espace libre dans un graphe [19]. On élabore par la suite des algorithmes de planification qui permettent d'exploiter ce modèle afin de générer une trajectoire praticable pour le robot [20].

1.3.3.2 Approche locale :

Le principe de l'approche locale consiste à déterminer les mouvements du robot mobile en ne considérant qu'une représentation locale de l'environnement. Dans ce type d'approches, un dispositif de perception est indispensable car on considère que le robot évolue dans un univers qui lui est totalement inconnu globalement [17].

1.3.3.3 Approche mixte ou hybride :

L'intérêt de cette approche est d'unifier des avantages des deux approches cités précédemment, à savoir :

-Temps de calcul faible pour l'approche locale.

-Chemin optimal généré par l'approche globale.

Le principe de cette approche est de générer un chemin optimal, en utilisant une méthode globale, donnant l'allure générale de la trajectoire à suivre. Ensuite, le robot exécutera cette trajectoire, à l'aide d'un générateur local de trajectoire et de son système de perception, tout en évitant de rentrer en collision avec des obstacles inattendus [17].

1.3.4 Navigation en environnement d'intérieur :

La navigation en milieu d'intérieur concerne principalement des environnements humains hautement structurés, tel que des bâtiments publics ou des ateliers. En plus, il existe plusieurs possibilités pour les représentations utilisées dans ce modèle du monde, parmi lesquelles on peut citer :

a) une représentation purement géométrique : la navigation par une carte métrique exploite une représentation géométrique du monde, qui peut être donnée par un utilisateur, ou construite par le robot lui-même, par différentes méthodes traitant le problème du SLAM (*Simultaneous Localization and Mapping)* [6].

b) une représentation topologique qui décrit les connexions entre différents lieux formant l'environnement : pour la génération d'une trajectoire entre deux lieux définis dans une carte topologique, il suffit de rechercher un chemin dans le graphe. Les consignes pour naviguer seront des commandes référencées capteurs.

1.3.5 Navigation en environnement d'extérieur :

La différence principale entre la navigation en extérieur et celle en intérieur, est que les conditions d'illuminations sont non contrôlables en extérieur ; quelques fois elles ne sont même pas prédictibles. Généralement, les environnements d'extérieur sont plus riches en informations (couleur, texture), mais aussi plus complexes. Pour faire face à cette plus grande complexité de l'environnement, il faut adopter des capteurs qui en donnent une description la plus riche possible : la vision s'impose généralement dans ce contexte, que cela soit en monoculaire [21], [22] ou en stéréovision [23].

Il existe plusieurs travaux pour la navigation en environnement d'extérieur, travaux assez différents les uns des autres du fait de la variabilité de ces environnements. Cela va de la navigation de voitures sur route ou en milieu urbain, jusqu'à la navigation d'un robot d'exploration planétaire [27].

Essentiellement il faut garantir que le véhicule reste:

- Sur une route.
- Ou sur une voie de circulation.
- Ou à distance réglementaire du véhicule en avant ou en arrière et qu'il détecte tout autre objet sur la route.

Les scènes à traiter sont dynamiques (trafic, routier) et l'environnement structuré (marquages sur la chaussée). Dans ce contexte, le suivi visuel concerne la chaussée, pour un contrôle automatique du véhicule, pour évaluer les trajectoires [28].

La figure 1.4 représente les deux milieux d'extérieur (structuré et non structuré).

(a) (b)

Figure 1.4 : Navigation en milieu d'extérieur. a) structuré, b) semi structuré.

1.3.5.1 Navigation en environnement d'extérieur non structuré:

Les environnements non structurés manquent de primitives géométriques régulières (lignes blanches bien délimitées, largeur de la voie relativement constante, etc.) qui sont exploitées dans les approches décrites ci-dessus pour effectuer la navigation. Ce type de milieu contient plutôt des chemins de terre de la campagne, des terrains accidentés ou n'importe quelle zone traversable par un véhicule (robot) tout-terrain.

Dans les applications terrestres, le robot exécute généralement une tâche prédéfinie, comme le suivi d'un élément qu'il doit reconnaitre dans l'environnement tout au long d'une région navigable [25]. La détection des zones navigables [25] et la détection d'obstacles lors du déplacement du robot exploitent la construction et la fusion des cartes de traversabilité. Pour pouvoir exécuter une tâche de navigation, le robot requiert des fonctions additionnelles pour la détection et le suivi d'amers (discontinuité sur la ligne d'horizon, bâtiments, un grand arbre, etc.) exploités pour se localiser ou pour exécuter des commandes asservies.

Dans la section suivante nous présentons les travaux de navigation visuelle les plus pertinents, adaptées aux environnements structurés.

1.3.5.2 Navigation en environnement d'extérieur structuré:

La navigation en environnement structuré d'extérieur concerne les techniques de suivi de routes [26] ou de chemins en terrain plat. Dans l'état de l'art actuel, le succès du suivi de routes repose sur la bonne détection et estimation des marquages (lignes blanches, bords de la chaussée) et sur leur interprétation pour établir sur quelle voie navigable se trouve le robot. Dans ces systèmes, les modèles de l'environnement ne sont pas exagérément complexes grâce à l'utilisation d'informations géométriques du chemin.

Par analogie, la navigation sur des routes est similaire à la navigation dans des couloirs pour les environnements d'intérieur à l'exception des problèmes causés par les ombrages, des conditions d'illumination changeantes [37].

Le module de vision fournissait une description du chemin en considérant que le véhicule était devant la zone navigable. Cette description était complétée par des informations telles que la position, la vélocité, le cap, etc. Pour mieux localiser le chemin en cherchant uniquement dans une zone de l'image. Enfin, un modèle global de la scène était obtenu. Le bon fonctionnement du processus est vérifié par l'évaluation successive des modèles de la scène calculés sur les images précédentes, cela afin d'assurer le lissage et la continuité des contours du chemin.

1.3.6 Notre approche de navigation:

Une tâche principale à l'autonomie des robots est de déplacer en toute sécurité dans un environnement quelconque. Les informations extraites des capteurs embarquées sont utilisées pour construire des modèles représentatifs par des amers naturels trouvés dans la scène observée [3] par le robot mobile.

Nous nous intéressons à la navigation de Robucar dans un environnement d'extérieur structuré en utilisant une seule caméra. La navigation est réalisée à partir des données issues de cette caméra embarquée en se basant sur les techniques de détections des zones navigables basées elles aussi sur les méthodes de segmentation par contours actifs (snakes). L'extraction des zones navigables revient à extraire leur contour. L'ensemble de ce module visuel partant de l'acquisition d'une image à niveau de gris et finissant par l'extraction des contours des régions associées à ces zones. Pour la navigation, nous identifions (montrons) dans chaque image, la région sur laquelle est le robot, et nous exploitons les frontières qui limitent cette région, pour estimer dans cette image, une trajectoire à suivre.

1.4 Conclusion :

Dans ce chapitre nous avons présenté d'une manière générale le contexte de notre travail, la navigation des robots mobiles. Où nous avons commencé par citer la définition et les caractéristiques d'un RMA et sa capacité de réaliser des tâches sans interventions humain ou sous contrôle réduit, grâce à la perception qui est assurée par des capteurs. Ainsi, nous avons donné quelques exemples des RMA, puis parlé de la perception et les capteurs utilisés tels que les capteurs proprioceptifs (odomètre, gyroscope, etc.), et les capteurs extéroceptifs (télémètres, etc.).

Par la suite, nous avons rentré dans la problématique de notre travail qui est la navigation des RMA dans un environnement extérieur structuré, où nous avons commencé par les étapes de navigation qui sont : perception et modélisation, décision et planification et l'exécution ; ainsi que les approches de navigation (globale, locale, mixte). Ensuite, nous avons cité les modes de navigation : navigation en environnement d'intérieur (par carte métrique, par carte topologique) et la navigation en environnement d'extérieur (environnement structuré tels que les routes et les chemins godronnés, environnement non structuré comme des chemins de terre ou des terrains accidentés).

Enfin, nous avons parlé de notre approche développée traitant la navigation d'un RMA de type voiture Robucar pour le transport urbain dans un environnement d'extérieur structuré de type route. Cette approche consiste à suivre des zones navigables en temps réels à l'aide d'une caméra embarquée.

Dans le prochain chapitre nous nous rappellerons quelques notions du traitement d'image en mettant l'accent sur la méthode utilisée dans ce travail, la segmentation d'images par des contours actifs.

Chapitre1: *Navigation Des Robots Mobiles Autonomes*

Chapitre2 *: Traitement d'Image et Contours actifs*

Dans ce chapitre, nous présentons les définitions de base utilisées dans notre travail, où nous citons quelques notions fondamentales sur le traitement d'image ; nous commençons d'abord par quelques définitions concernant l'image et ses caractéristiques, après nous parlons du traitement d'image de base et les opérateurs connus tels que (réduction de bruit, rehaussement de contraste, etc.). Nous rappelons aussi les approches de segmentations largement utilisées dans la littérature.

Afin de ce chapitre, nous traitons la méthode de segmentation utilisée dans notre travail: les contours actifs en citant les domaines d'applications, et le principe de fonctionnement ainsi que les limitations et les inconvénients.

2.1 Introduction:

Les images constituent l'un des moyens les plus importants qu'utilise l'homme pour communiquer et transmettre le savoir et l'information depuis l'aube de l'humanité, dans la mesure ou une image à elle seule peut englober une quantité énorme d'informations. Il inventa des méthodes aussi diverses les unes des autres pour pouvoir communiquer. La perception visuelle était son outil le plus exploité, et le moyen de communication le plus utilisé était de loin le dessin.

Le domaine du traitement d'image connaît une progression importante liée à l'évolution des technologies de l'information. Dans de nombreux domaines, il est fait appel aux techniques d'analyse d'images (échographie, compression, reconnaissance de caractères, reconnaissance des visages, etc.) car, souvent ces images ne peuvent être étudiées telles quelles et doivent subir une étape de transformation nécessaire à l'augmentation de l'efficacité du traitement.

Un système de traitement d'images se compose essentiellement des fonctions suivantes : acquisition d'images, un prétraitement pour la réduction du bruit, analyser l'image pour arriver à une description synthétique de l'information brute contenue dans l'image.

2.2 Définitions et traitements de base :

2.2.1 Définition d'une image :

Une image est un support d'informations présentant les éléments d'une scène qui a été capturé par un appareil. En traitement de signal, une image est définit comme étant un signal bidimensionnel. Alors une image est définie par le nombre de pixels qui la compose en largeur et en hauteur ; l'étendue des teintes de gris ou des couleurs que peut prendre chaque pixel [28]. Nous pouvons classer les images sous trois types :

- **Image couleur :**

Ce type d'image comprend 03 plans d'images (RVB; rouge vert et bleu, ou ITS; intensité, teinte et saturation).Chaque image (R, V ou B) est codée sur 8 bits. La combinaison de ces trois plans d'images par synthèse additive donne l'image couleur [29] (voir la figure 2.1).Une image couleur occupe 3 fois plus d'espace mémoire qu'une image à niveau de gris (1 pixel->1 octet) [29].

1 pixel->1 octet pour le rouge + 1 octet pour le vert + 1 octet pour le bleu.

Figure 2.1 : Image couleur.

- **Image à niveau de gris (NDG) :**

L'intensité du pixel f(x,y) de cette image varie entre un Gmin et Gmax , ces deux valeurs appartiennent à l'intervalle [0,255] (voir la figure 2.2).

L'image comprend un nombre de niveaux de gris supérieur à 2 [29].

Figure 2.2: Image à niveaux de gris.

L'opération de conversion en NDG peut se faire à l'aide de l'équation suivante:

$$f(x,y) = 0.178 \times fR(x,y) + 0.818 \times fV(x,y) + 0.016 \times fB(x,y)$$

- **Image binaire (noir et blanc) :**

Dans ce type d'image l'intensité du pixel f(x,y) est égale à 0 ou à 255 correspondants au noir ou blanc respectivement comme l'indique la figure 2.3 .Chaque pixel est codé sur 1 bit. Une image binaire occupe 8 fois moins d'espace mémoire qu'une image à NDG [29].

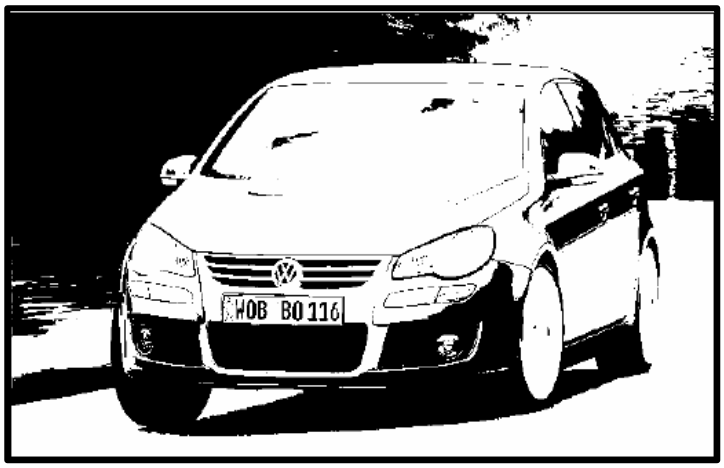


Figure 2.3: Image binaire (noir et blanc).

2.2.2Caractéristiques d'une image :

2.2.2.1 Pixel :

Une image est constituée d'un ensemble de points appelés pixels ; où le pixel représente ainsi le plus petit élément constitutif d'une image numérique auquel on peut associer individuellement une couleur (ou un NDG) et une intensité. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image finalement obtenue [4].

Chaque pixel est défini dans une image par des coordonnées et étant donné que l'écran effectue un balayage de gauche à droite et de haut en bas, comme illustre-la figure 2.4 ; on désigne généralement par les coordonnées [0,0] le pixel situé en haut à gauche de l'image.

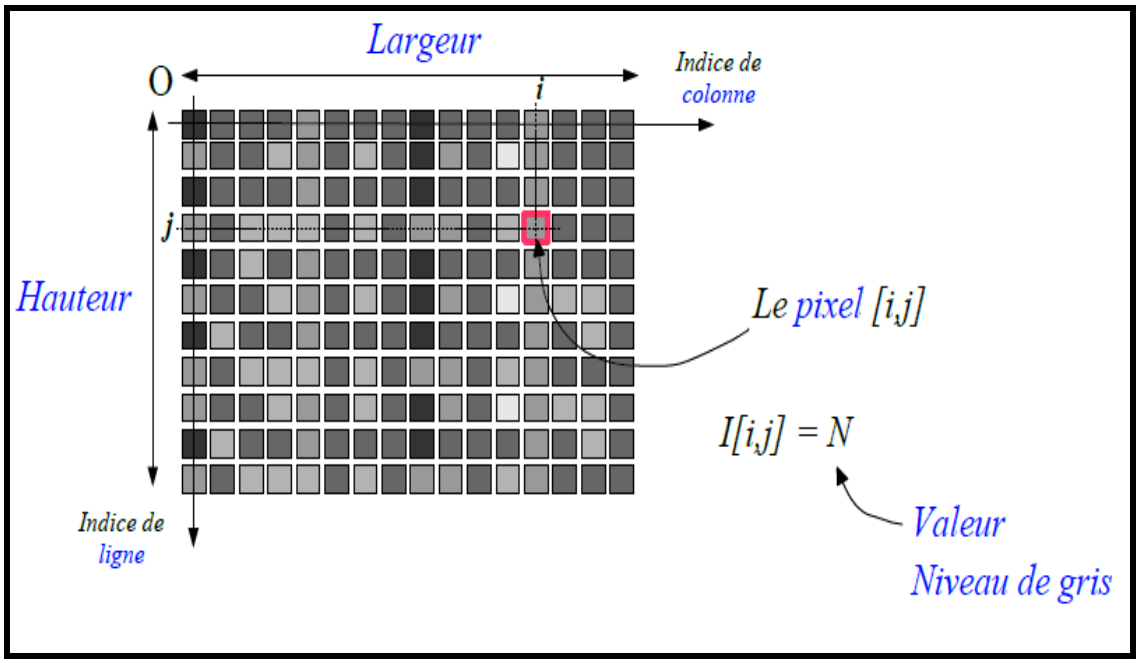


Figure 2.4 : Pixels d'une image.

2.2.2.2 Définition :

On appelle définition le nombre de points (pixels) constituant l'image, c'est-à-dire, sa «dimension informatique» (le nombre de colonnes de l'image que multiplie son nombre de lignes). Une image possédant 640 pixels en largeur et 480 en hauteur aura une définition de [4] :640 pixels par 480, notée 640x480.

2.2.2.3 Résolution :

La résolution, détermine par contre le nombre de points par unité de surface, exprimée en points par pouce (PPP, en anglais DPI pour Dots Per Inch); un pouce représentant 2,54 cm. La résolution permet ainsi d'établir le rapport entre le nombre de pixels d'une image et la taille réelle de sa représentation sur un support physique.

Une résolution de 300 dpi signifie donc 300 colonnes et 300 rangées de pixels sur un pouce carré ce qui donne donc 90000 pixels sur un pouce carré [4].

2.2.2.4 Niveau de gris :

Dans une image en NDG (nuances de gris), la couleur d'un pixel peut prendre des valeurs allant du noir (0) au blanc (255) en passant par un nombre fini de niveaux intermédiaires obtenus par dégradation du noir. Le pixel est ainsi codé sur un octet (sur 8 bits). Nous donnons un exemple de dégradation de niveaux de gris dans le tableau qui suit :

Couleur	BLANC	GRIS 12.5%	GRIS 25%	GRIS 37.5%	GRIS 50%	GRIS 62.5%	GRIS 75%	GRIS 87.5%	NOIR
Niveau de gris	255	224	192	160	128	96	64	32	0

Tableau 2.1 : Exemple de NDG d'une image.

2.2.2.5 Histogramme :

On appelle histogramme de l'image I, la fonction H définie sur l'ensemble des entiers naturels par: H(x) = Card {P : I(P) = x}. Ou H(x) traduit le nombre d'apparitions du niveau de gris ***x*** dans l'image I. L'histogramme est un outil privilégie en analyse d'images car il représente un résumé simple, mais souvent suffisant du contenu de l'image [4]. Ainsi, l'histogramme d'une image en 256 NDG sera représenté par une graphique possédante 256 valeur en abscisses, et le nombre de pixels de l'image en ordonnées.

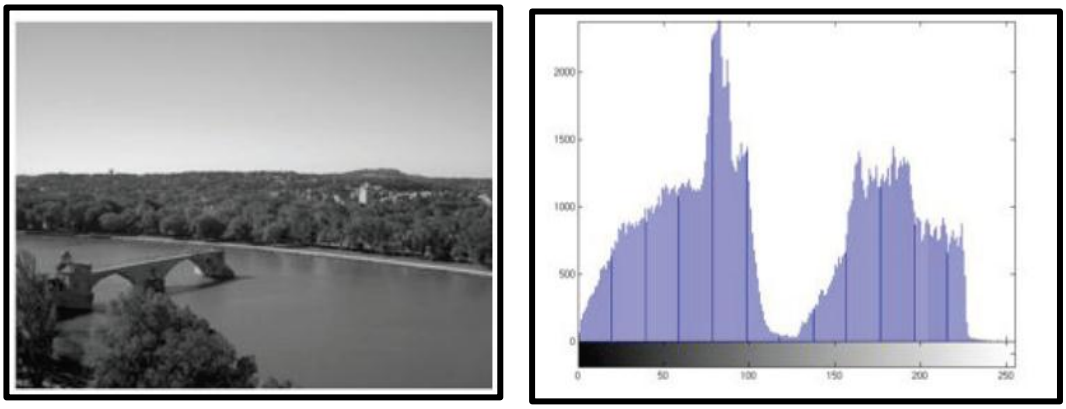


Figure 2.5 : Image et son histogramme.

2.2.3 Prétraitements :

2.2.3.1 Introduction :

Les prétraitements permettent la préparation des données reçues des capteurs à la phase suivante d'analyse consacrée à l'extraction des paramètres. Cette phase n'est possible et surtout fiable que si les données des capteurs sont dénuées de bruit, corrigées de leurs erreurs éventuelles, homogénéisées et réduites à l'essentiel [4].

Le prétraitement a pour but de :

-Elimination du bruit : plusieurs filtres ont été implémentés comme les filtres linéaires et les filtres non linéaires.

-Renforcement de contraste et l'homogénéité des régions grâce aux méthodes de modification d'histogramme et de rehaussement de contraste.

2.2.3.2 Opérateurs de prétraitement :

Les opérateurs de prétraitement sont divisés en trois catégories [29] :

- Opérateurs de réduction du bruit.

- Opérateurs de rehaussement du contraste.

- Opérateurs de modification d'histogramme.

2.2.3.2.1 Réduction de bruit (filtrage) :

L'image possède dans sa nature une redondance spatiale en information ; les pixels voisins en 4-voisins ou en 8-voisins possèdent les mêmes caractéristiques du point de vue niveau de gris. Une présence de bruit dans une image provoque une variation brutale de niveau de gris d'un pixel par rapport aux autres voisins [30].

Le principe de filtrage est de modifier la valeur des pixels d'une image, généralement dans un but d'améliorer son aspect. En pratique, il s'agit de créer une nouvelle image en se servant des pixels de l'image d'origine. Sa qualité sera améliorée [31]. Le filtrage manipule uniquement les données de l'image numérisée. On ne fait aucune supposition sur ce que représente l'image. Le résultat obtenu suite à un filtrage dépend donc énormément de la qualité du signal de l'image d'origine [31].

Les opérations de réduction du bruit (filtrage) sont divisées en deux catégories : les filtres linéaires (moyen, gaussien, etc.) et les filtres non linéaires (médian, min max, etc.). Le type de filtrage le plus simple et facile à implémenter est le filtrage linéaire [30].

Dans ce que suit, nous donnons des exemples des filtres linéaires :

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \qquad \frac{1}{9}\times\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Filtre gaussien 3x3. Filtre moyen 3x3.

2.2.3.2.2 Rehaussement de contraste :

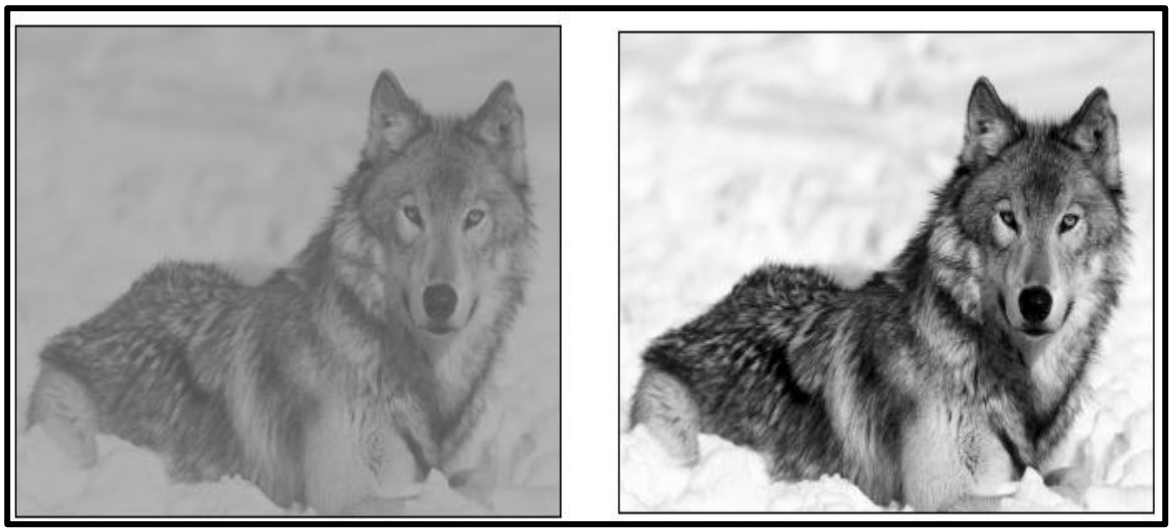
Pour un certain type d'images, il arrive que la transition entre les régions soit floue ; l'objectif cherché par le rehaussement du contraste est de diminuer l'étendue de la zone de transition sans affecter l'intensité moyenne des régions [37].

Cela est réalisé par un produit de convolution entre l'image et la matrice suivante:

$$\begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

Masque de contraste.

Comme illustre la figure 2.6 :



(a) (b)

Figure 2.6 : a) image à faible contraste, b) image à fort contraste.

2.2.3.2.3 Modification d'histogramme :

La modification d'histogramme est une fonction qui fait correspondre à chaque NDG d'un pixel de l'image originale un autre NDG. Chaque NDG est modifié dans le but d'accroître le contraste. La modification de l'histogramme est adoptée lorsqu'une majeure partie des NDG présents dans l'image est concentrée dans un faible intervalle sur l'échelle des intensités lumineuses [4].

Elle contient l'égalisation qui a pour but d'harmoniser la répartition des niveaux de luminosité de l'image, l'élargissement d'histogramme (expansion de la dynamique) consiste à répartir les fréquences d'apparition des pixels sur la largeur de celui-ci.

Le recadrage de la dynamique consiste à établir les fréquences d'apparition des pixels sur la longueur d'histogramme, et enfin l'extraction d'une fenêtre d'intensité ce qui est l'inverse du recadrage dynamique.

2.2.4 Seuillage :

Cette fonction consiste à examiner les pixels et à ne retenir que ceux dont la valeur est comprise entre deux seuils donnés. Ceci a pour but de sélectionner des plages de NDG correspondant à des régions d'intérêt dans la scène étudiée. Selon le cas, l'utilisateur peut affecter un code différent à chaque région ou obtenir une image binaire où le noir représente toutes les régions (les objets) et le blanc représente le fond. Dans ce dernier cas, l'opération de seuillage s'appelle ***binarisation***. Alors, le seuillage permet de mettre en évidence (détecter, localiser) les formes ou les objets.

2.3 Segmentation d'Images:

2.3.1 Définition :

La segmentation d'images peut être considérée comme un traitement de bas niveau. Elle a pour but de permettre l'extraction d'éléments de l'image. Elle n'est généralement qu'une première étape essentielle dans le processus d'interprétation d'une scène. Elle effectue un partitionnement de l'image en un certain nombre de régions disjointes. La segmentation peut être liée directement au NDG de chaque pixel, ou bien à l'attribut estimé dans le voisinage du pixel, tel que la valeur moyenne, la variance ou des paramètres de texture plus complexes. Alors, l'objectif général de segmentation est d'extraire les informations pertinentes (essentiels) dans l'image.

2.3.2 Approches de segmentation :

Lors d'une étude sur une image, l'objectif est de rechercher une particularité dans l'ensemble de l'image ou dans une partie de l'image. Pour une application particulière, il s'agit d'extraire les informations pertinentes [33]. Il est reconnu deux grandes approches : l'approche région et l'approche frontière (contour).

Ces deux approches sont duales car une région définit une ligne sur son contour et une ligne fermée définit une région intérieure.

II.3.2.1 Approche frontière (contour) :

Dans une image, les variations d'intensité représentent des changements de propriétés physiques ou géométriques de la scène ou des objets observés. Elles peuvent correspondre, par exemple, à des variations d'illumination, à des ombres, à des changements d'orientation ou de distance par rapport à l'observateur ou encore à des variations d'absorption des rayons (lumineux, X, etc.).

Dans un grand nombre de cas, ces variations d'intensité sont de informations importantes qui permettent de constituer les frontières des régions correspondant à des bords ou des parties d'objets de la scène, d'où le nom donné à cette approche de la segmentation [33].

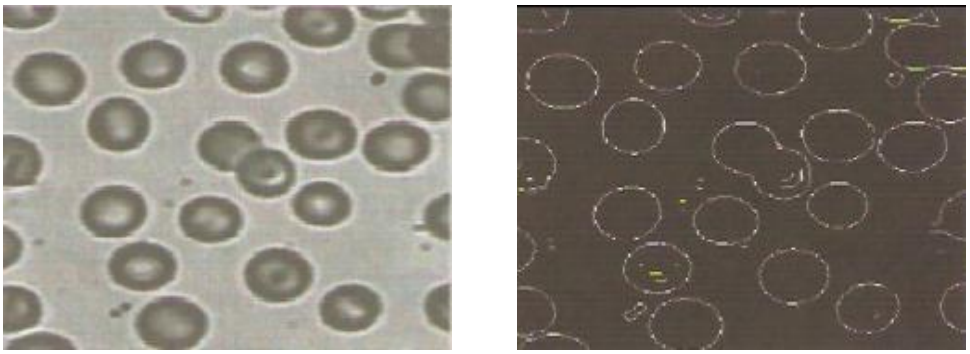
L'approche classique de détection de contours consiste à balayer l'image avec une fenêtre définissant une zone d'intérêt. A chaque position, un opérateur est appliqué sur les pixels de la fenêtre afin d'estimer s'il y a une transition significative au non.

D'une manière générale, la segmentation par approche frontière consistera en :

- Une détection des contours,
- La fermeture des contours,
- Le codage des contours.

En d'autres termes, un contour est défini comme le lieu de variation significatif de l'information. Par conséquent, trouver les contours dans une image revient à évaluer la variation du NDG en chaque pixel de l'image.

La figure 2.7 illustre une détection de contour par une méthode dérivative.



(a) (b)

Figure 2.7: Exemple de détection de contour.

a) Image originale, b) Image contour.

2.3.2.2 Approche région :

Dans l'approche région, c'est la similitude des points connexes qui est favorisée. Cela peut être vu comme une technique contextuelle. Les points connexes ayant des propriétés similaires (attributs) : intensité de gris, couleur, texture, vont être réunis dans le même ensemble. COCQUEREZ dans [32] présente des attributs stochastiques, fractales, surfaciques, géométriques, etc. l'algorithme généralement utilisé procède par croissance de région [32]. Cette technique consiste à regrouper des points voisins pour former des régions de plus en plus importantes.

2.3.3 Détection de contours :

La détection de contours dans les images a débuté de façon extrêmement empirique par des opérateurs locaux qui, soit estimaient un gradient, soit convolaient l'image par des masques caractéristiques des contours [34]. Dans les années 80, des approches plus systématiques ont été mises en place par Marr [35], puis Canny [36], pour obtenir des contours plus significatifs. Ces travaux ont abouti à une bonne compréhension de ce qu'il faut faire pour détecter les contours, mais la définition même des contours demeure très vague, ce qui rend ces techniques encore peu efficaces sur un problème

concret. Par définition, un contour est une brusque variation du NDG dans une image d'une amplitude **a** et avec une pente **p**. Un contour peut être défini comme une "*marche d'escalier*" si le contour est *net*, comme une "*rampe*" si le contour est *plus flou* ou comme un "*toit*" s'il s'agit d'une *ligne sur un fond uniforme* [36].

Les seuls modèles de contours utilisables sont ceux de contours idéalisés, comme ceux présentés sur la figure suivante. Le plus utilisé est celui en marche d'escalier.

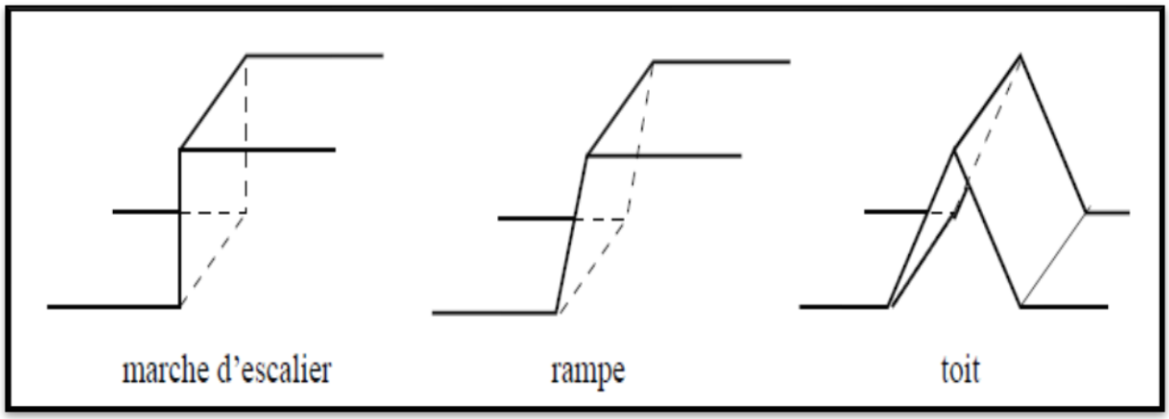


Figure 2.8 : Quelques modèles de contours.

2.3.3.1 Approches dérivatives pour la détection de contours :

❖ **Approche gradient :**

Cet opérateur permet de caractériser et de repérer les zones de transition. La détection de contours revient à déterminer les extrema locaux dans la direction du gradient [36].

Le gradient d'une image est obtenu en appliquant le vecteur suivant :

$$\nabla I(x,y) = \left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right) \quad (2.1)$$

a) Roberts :

Le détecteur de Roberts permet de calculer le gradient bidimensionnel d'une image de manière simple et rapide. Il amplifie les zones où la norme du gradient spatial est importante qui correspond souvent aux contours. Ce principe ne diffère pas beaucoup de celui des opérateurs de Prewitt et Sobel. Ces masques répondent maximalement aux contours orientes à 45°par rapport à la grille de pixels [32].

Le masque de Roberts est donné par :

1	0
0	-1

0	1
-1	0

Masque de convolution de Roberts.

b) Sobel et Prewitt :

Les opérateurs de Sobel et de Prewitt permettent d'estimer localement la norme du gradient spatial bidimensionnel d'une image en NDG. Ils amplifient les régions de fortes variations locales d'intensité correspondant aux contours. Ces opérateurs consistent en une paire de masques de convolution 3×3 [32]. L'application séparée de chacun des masques donne une estimation des composantes horizontales et verticales du gradient par un simple filtrage linéaire avec un masque 3×3. Il est ensuite possible de calculer la norme et la direction du gradient en chaque point à partir des composantes du gradient ∇x et ∇y.

La norme du gradient ainsi estimée correspond à l'intensité attribuée au pixel courant. C'est donc l'image de la norme du gradient que l'on visualise généralement [37].

Nous donnons ici les masques de Sobel et de Prewitt :

1	0	-1
1	0	-1
1	0	-1

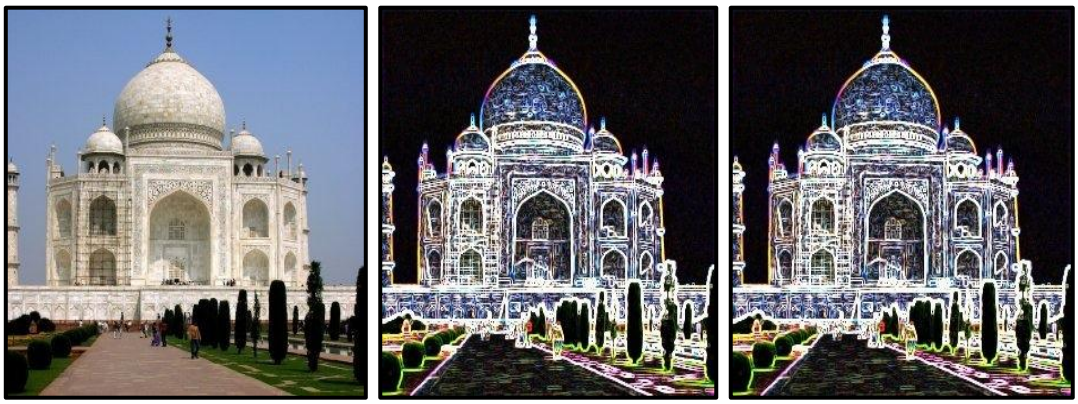
1	1	1
0	0	0
-1	-1	-1

Masque de convolution de Prewitt.

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Masque de convolution de Sobel.



(a) (b) (c)

Figure 2.9 : Exemples de gradient.

(a) : Image originale, (b): Contour par Sobel, (c) : Contour par Prewitt.

❖ **Approche Laplacien :**

On vient de voir des filtres basés sur la recherche de maxima de la dérivée première. Les filtres larges qu'on va maintenant étudier recherchent les zéros de la dérivée seconde (méthode permettant de bien mettre en évidence les maxima de la dérivée première) ou, plus précisément, du Laplacien qui est une dérivation au deuxième ordre [32].

$$\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \qquad (2.2)$$

Dans le cas d'une image, il n'existe pas une dérivée seconde unique mais quatre dérivées partielles (Selon x^2, y^2, xy et yx).

En pratique, on enlève cette ambiguïté en ayant recours à l'opérateur Laplacien qui fait la somme des deux dérivées partielles principales seulement [37].

Les considérations concernant le bruit dans la dérivée première sont encore plus importantes dans les calculs de la dérivée seconde. On utilise donc couramment une combinaison de lissage et Laplacien ce qui correspond au Laplacien d'une Gaussienne.

L'estimation de la dérivée seconde étant très sensible aux bruits, il convient de filtrer très fortement l'image avant de mesurer le Laplacien. Ainsi, afin de limiter les réponses dues au bruit de l'image I, le plus souvent [37].

2.3.3.2 Dérivation par filtre optimale :

Le filtrage optimal repose sur la définition des critères d'optimalité de la détection de contours. Ces critères débouchent sur des filtres de lissage optimaux [35].

Parmi ces filtres, nous citons le filtre de Canny et le filtre de Deriche.

La figure 2.10 montre des exemples de contours par Canny et Deriche.

(a) (b) (c)

Figure 2.10: Exemples de détection contours.

a) Image originale, b) Contour par Deriche, c) Contour par Canny.

2.4 Contours actifs (Snakes) :

2.4.1 Introduction :

Les contours actifs (snakes) sont largement utilisés en segmentation d'images pour leur capacité à intégrer les processus de détection et de chaînage des contours en seul processus de minimisation d'énergie. Toutefois l'estimation des paramètres et les problèmes d'initialisation font des contours actifs une méthode difficile à calibrer. Le modèle de contours actifs se présente sous la forme d'une courbe fermée ou ouverte dont l'initialisation est située à proximité du contour qu'on veut obtenir et dont l'évolution s'effectue selon un processus itératif de déformation contrôlé par un test de convergence.

Les snakes tiennent leur nom de leur aptitude à se déformer comme des serpents. Depuis la publication de l'équipe Kass, Witkin et Terzopoulous [5], les modèles déformables sont devenues un sujet très important pour la communauté de traitement d'images. Dans ce modèle le processus de déformation correspond à la minimisation d'une fonctionnelle énergie entre la courbe et l'image dans laquelle nous cherchons à extraire les objets.

Les domaines d'utilisations sont nombreux tant en 2D qu'en 3D; tels la reconnaissance de formes, la simulation, le suivi de scènes, la segmentation d'images, la robotique mobile, segmentation et reconnaissance des formes en 3D, le suivi de contours ou de mouvement (tracking) [38].

***Définition du snake:**

Les contours actifs sont modélisés par une courbe paramétrique continue fermée ou non qui se déforme en fonction de différentes énergies pour venir se positionner sur les frontières de l'objet. Pour les images numériques, le traitement s'effectuera avec un certain nombre de points chaînés, et dont le nombre varie en fonction de la précision désirée.

Nous venons voir que les contours actifs (snakes) permettent de résoudre un problème par minimisation d'une fonctionnelle d'énergie .Cette fonctionnelle représentera alors l'ensemble des énergies, auxquelles sont appliquées leur coefficients respectif. L'approche variationnelle tendra à minimiser cette fonctionnelle d'énergie.

2.4.2 Principe des contours actifs :

Le principe des contours actifs est de faire évoluer un contour initial autour de l'objet d'intérêt vers une position d'équilibre, c'est-à-dire une direction des bords de l'objet à détecter. Entre deux itérations, la vitesse des points est régie par une équation mettant en jeu des forces à appliquer au contour .Ces forces dépendent des données de l'image (gradient et l'intensité) et des propriétés du contour (rigidité, élasticité, etc.). La position d'équilibre du contour peut être prise comme étant le minimum d'une fonction d'énergie dont la dérivée correspond aux forces à appliquer [5].

Le principe des contours actifs est illustré par la figure suivante :

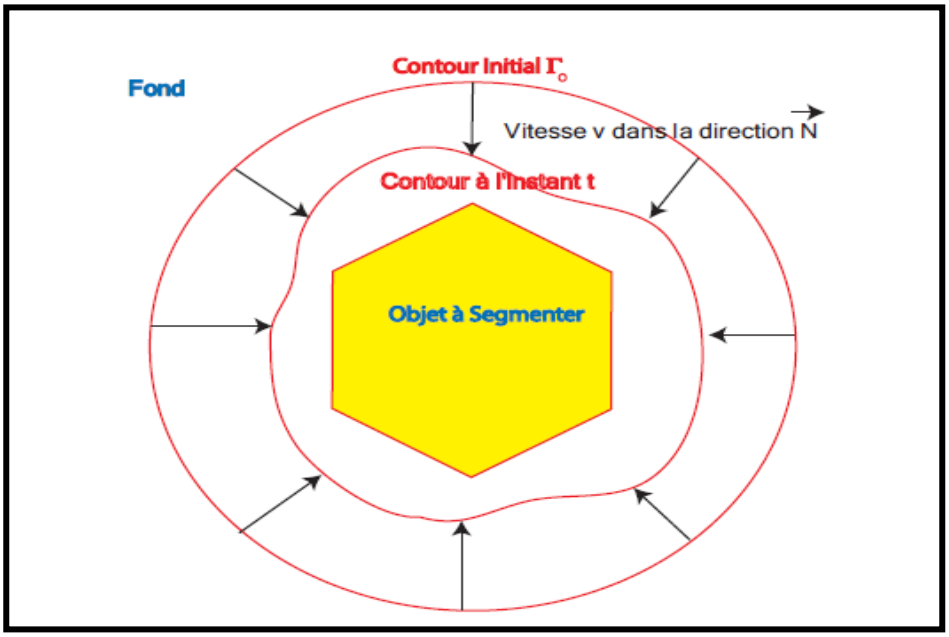


Figure 2.11 : Principe des contours actifs.

Nous venons de voir que les contours actifs permettent de résoudre un problème par minimisation d'une fonctionnelle d'énergie [39]. Nous allons maintenant examiner les différentes énergies en jeu.

2.4.3 Energies:

La fonctionnelle d'énergie attachée au contour actif est composée de trois énergies [38] :

-Energie interne.

-Energie externe.

-Energie de contexte.

$$\boldsymbol{E}(\boldsymbol{v}(\boldsymbol{s})) : \rightarrow E_{interne}\,(v(s)) + E_{externe}(v(s)) + E_{contexte}\,(v(s))$$

Avec: v(s) représente la courbe initiale (Snake) ; et s l'abscisse curviligne.

Nous allons étudier par la suite en détail ces trois types d'énergies.

2.4.3.1 Energie interne :

L'énergie interne est propre au contour actif, elle ne dépend que de points de snake. C'est elle qui gère la cohérence de la courbure. Elle maintient la cohésion des points et la raideur de la courbure, qui est utilisée comme terme de régularisation.

$$E_{intèrne} = \int_0^1 \left(\frac{\alpha}{2}(\boldsymbol{s})\left|\left|\boldsymbol{v}'(\boldsymbol{s})\right|\right|^2 + \frac{\beta}{2}(\boldsymbol{s})\left|\left|\boldsymbol{v}''(\boldsymbol{s})\right|\right|^2\right)\boldsymbol{ds} \qquad (\mathbf{2.3})$$

Les termes *v'* et *v''* sont les dérivées première et seconde de *v* par rapport à *s*. Le terme du *1er* ordre correspond à la tension. Il prend une valeur importante quand la courbe se distend. Lorsque $\alpha = 0$, la courbe peut présenter des discontinuités. On parlera donc d'*énergie de continuité.*

Le terme de *2ème* ordre correspond à la *courbure*. Il prend une valeur importante lorsque la courbe s'incurve rapidement c'est-à-dire pour l'obtention de coins. Lorsque β = *0*, la courbe peut prendre une forte convexité, par contre lorsque β est grand, la courbe tendra vers un cercle si elle est fermée, ou une droite si elle est ouverte [38].

Les paramètres *α* et β devant chaque terme sont les poids de la fonction; généralement ils sont fixés au début du processus de déformation pour tous les points de contour. Ces paramètres ont une grande importance sur la performance de ce processus.

2.4.3.2 Energie externe :

L'énergie interne qu'on vient d'analyser, gère la régularisation du contour actif. L'énergie externe correspond à l'adéquation aux données. Cette énergie externe prend en compte les caractéristiques de l'image. On rappelle ici que ce sont les contours de formes qui sont recherchés donc les points de fort gradient ou des points ayant une propriété de position par rapport à une couleur donnée [38].

2.4.3.2.1 Gradient :

L'énergie de Gradient est la deuxième énergie du contour actif qui dépende de l'image. Cette énergie externe est d'une importance première pour la détection du contour. En effet, un contour est généralement caractérisé par une forte différence entre les valeurs de plusieurs pixels. Ainsi, si la dérivée d'une fonction représente les pentes d'une courbe, le gradient montrera les fortes différences des contours de l'image.

Pour la recherche des zones de fort contraste dans l'image, on introduit la fonction suivante [38]:

$$E_{externe} = -\int_0^1 ||\boldsymbol{\nabla I}(\boldsymbol{v}(\boldsymbol{s}))||^2 \boldsymbol{ds} \qquad (\mathbf{2.4})$$

Où $\nabla I\,(v(s))$ représente le gradient de l'image *I* en *v*(*s*).

2.4.3.2.2 Intensité:

Cette énergie, au contraire, permet de sélectionner les zones sombres ou claires selon le signe choisi.

$$\boldsymbol{E_{intensite}} = \pm \int_0^1 (\boldsymbol{I}(\boldsymbol{v}(\boldsymbol{s})) - \boldsymbol{i_0})^2 \, \boldsymbol{ds} \qquad \textbf{(2.5)}$$

La valeur *i*0 peut être introduite ou non, un certain seuillage. On peut ainsi favoriser la position du contour dans une zone donnée.

2.4.3.2.3 GradientVectorFlow(GVF) :

Cheniang XU dans [40] constatant la médiocrité de la qualité de la convergence de la courbe de contour actif vers le contour souhaité dans les zones à forte concavité, introduit un nouveau potentiel dans [41]. Il s'agit d'une nouvelle force externe qui traduit la diffusion isotopique d'un flux externe. Il définit le champ "*Gradient Vector Flow*", le "GVF" comme le champ de vecteurs :

$$\boldsymbol{v}(\boldsymbol{x},\boldsymbol{y}) = [\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y})\boldsymbol{v}(\boldsymbol{x},\boldsymbol{y})]^{\mathrm{t}}$$

Qui minimise la fonctionnelle d'énergie.

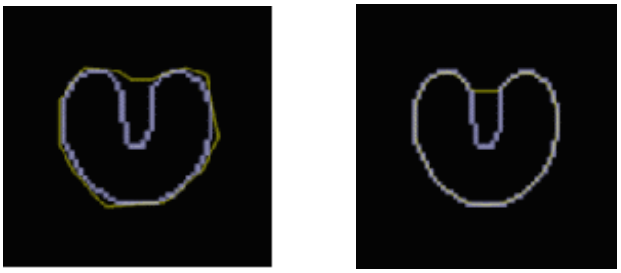


Figure 2.12 : Le contour actif avec la méthode de GVF.

XU et PRINCE dans [42] proposent une généralisation, le "*Generalized Gradient Vector Flow*", le "GGVF" [42].

Ce nouveau potentiel, d'un intérêt certain lorsque l'objet à segmenter est unique, peut poser problème dans le cas d'objets multiples dans des images réelles, la diffusion du gradient pouvant créer des interférences entre les zones d'influence des différents objets. Le temps de calcul de ce GVF peut aussi représenter un frein à cette méthode. Xu annonce pour une image 256 x 256 en 256 NDG un temps de 2 s en Matlab.

2.4.3.3 Energie de contexte :

L'énergie de contexte, parfois appelée énergie de contrainte, permet d'introduire des connaissances a priori sur ce qu'on cherche. Entre autres, on place, sous cette rubrique, l'énergie ballon introduite par Laurent D.Cohen [43]. Les contours actifs (*snakes*), de par leur discrétisation, ont une tendance naturelle à se rétracter. La minimisation de l'énergie implique une minimisation de distance. La force ballon va tendre à gonfler le contour actif ou accélérer sa rétractation selon le signe de la force introduite. De plus, cette force va permettre de dépasser les contours présentant un faible gradient et ainsi de sortir du bruit pour atteindre une frontière plus fortement marquée.

Il s'agit d'une force normale au contour en chaque point.

$$\boldsymbol{F_{ballon}}\big(\boldsymbol{v}(\boldsymbol{s})\big) = k\vec{\boldsymbol{n}}(\boldsymbol{s}) \qquad \textbf{(2.6)}$$

Ou : $\tilde{n}(s)$ est un vecteur unitaire normal à la courbe en v (s). L'intensité de l'énergie ballon est un scalaire généralement négatif (expansion du ballon) proportionnel à l'aire intérieure du contour.

2.4.4Méthodes d'optimisation :

Plusieurs méthodes ont été proposées pour résoudre le problème de minimisation de la fonctionnelle d'énergie. Nous allons présenter ici trois méthodes utilisées pour l'optimisation de la fonctionnelle d'énergie:

- **L'approche variationnelle classique** : qui est introduite par Kass [5] peut être la plus développée et la plus déclinée ; qui tire avantage des développements mathématiques de l'analyse numérique.
- **La programmation dynamique** : introduite par Amini [44] qui utilise les avancées de l'informatique.
- **L'algorithme Greedy** : proposé par Williams et Shah dans [45] qui est apprécié pour son temps de traitement plus rapide [46].

Avant d'examiner ces différentes méthodes, nous allons préciser comment les différences finies permettent de résoudre informatiquement le problème de l'implémentation de notions définies dans un espace réel.

2.4.4.1 Différences finies :

Les dérivées d'une fonction par rapport à une variable peuvent être approximées par des différences finies ou des éléments finis. Laurent David COHEN et Isaac COHEN [47] [48] ont montré que la recherche de solutions dans un espace de Sobolev était équivalente à celle dans un espace des fonctions polynomiales de dimension finie. Ils obtiennent, avec des éléments finis, des calculs similaires aux calculs utilisant les différences finies. Selon Laurent David Cohen, la méthode par éléments finis est moins coûteuse et plus stable.

II.4.4.1.1 Continuité:

L'énergie de continuité fait partie des énergies dites internes au contour actif. Cette énergie régit la distance entre les différents points du snake. Ainsi, lorsque le coefficient de l'énergie de continuité est nul, les points du snake pourront se déplacer aussi loin les uns des autres que l'image leur permettra. Dans le cas contraire, c'est-à-dire lorsque le coefficient de continuité est très élevé, le snake sera rigide. Elle représente l'élasticité de contour actif.

La dérivée première des coordonnées par rapport au paramètre *s* peut être approximée par différences finies et devient :

$$\|\acute{v}_i(s)\|^2 = \left\|\frac{dv_i}{ds}\right\|^2 \quad (2.7)$$

L'énergie de continuité est alors liée à :

$$\|v_i - v_{i-1}\|^2 = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 \quad (2.8)$$

2.4.4.1.2 Courbure :

L'énergie de courbure définit l'angle formé par trois points adjacents du snake. Plus le coefficient attribué à cette énergie sera fort, et plus le snake aura la forme connue pour être la forme la moins « coûteuse » en énergie .Elle représente la rigidité de contour.

La dérivée seconde s'approxime par :

$$\|v''_i(s)\|^2 = \left\|\frac{d^2v_i}{ds^2}\right\|^2 \quad (2.9)$$

L'énergie de courbure dépend alors de :

$$\|v_{i-1} - 2v_i + v_{i+1}\|^2 = (x_{i-1} - 2x_i + x_{i+1})^2 + (y_{i-1} - 2y_i + y_{i+1})^2 \quad (2.10)$$

2.4.4.2 Approche variationnelle :

Dans la méthode des contours actifs, il s'agit de minimiser une fonctionnelle d'énergie, composée d'une énergie interne, d'une énergie externe, éventuellement d'une énergie de contexte. Comme nous l'avons déjà précisé, la recherche du contour est limitée au cas d'une courbe plane [47].

Le contour initial *v*(0) est défini par l'utilisateur et la courbe évolue avec une certaine vitesse. Le problème est de trouver cette vitesse telle que la courbe évolue vers un minimum local correspondant aux contours des objets ou régions à segmenter [49].

La courbe subit, par exemple à deux types de forces :

-Des forces intérieures qui imposent une certaine régularité. Le coefficient α impose l'élasticité et le coefficient β impose la rigidité de la courbe.

-Une force extérieure –Force d'image (terme de potentiel) [50] qui pousse la courbe vers les zones qui correspondent aux attributs recherchés.

$$P(\boldsymbol{v}) = -\|\boldsymbol{\nabla I}(\boldsymbol{v})\|^2 \qquad (2.11)$$

La courbe est attirée par le minimum local du potentiel, c'est-à-dire les maxima locaux du gradient, donc des contours. Cette équation peut avoir plusieurs solutions puisque l'énergie peut avoir plusieurs minima locaux. La solution que l'on cherche est localisée dans une région donnée et on suppose posséder une valeur approchée de la solution V^0.

2.4.4.3 Programmation dynamique :

La programmation dynamique est une méthode classique de résolution de problèmes d'optimisation. Son application aux contours actifs est due à Amini, Weymouth et Jain [44]. Cette approche peut être une alternative intéressante au calcul variationnel.

Amini considère l'équation classique :

$$\boldsymbol{E_{tot}} = \int_0^1 \boldsymbol{E_{ext}}\left(\boldsymbol{v(s)}\right) + \frac{1}{2}\left(\boldsymbol{\alpha(s)}|\boldsymbol{v_s(s)}|^2 + \boldsymbol{\beta(s)}|\boldsymbol{v_{ss}(s)}|^2\right) ds = \int_0^1 \boldsymbol{E_{ext}} + \boldsymbol{E_{int}} \qquad (2.12)$$

En représentant la fonction à intégrer par *F*(*s,vs,vss*), où en discrétisant avec :

$$\boldsymbol{E_{int}(v_i)} = \left(\boldsymbol{\alpha_i}|\boldsymbol{v_i} - \boldsymbol{v_{i-1}}|^2 + \boldsymbol{\beta_i}|\boldsymbol{v_{i+1}} - 2\boldsymbol{v_i} + \boldsymbol{v_{i-1}}|^2\right)/\boldsymbol{2} \qquad (2.13)$$

Après la discrétisation, on convient:

$$\boldsymbol{E_{tot}} = \sum_{i=0}^{n-1} \boldsymbol{E_{int}}\left(\boldsymbol{v_i}\right) + \boldsymbol{E_{ext}}(\boldsymbol{v_i}) \qquad (2.14)$$

Cette somme peut être vue comme un processus discret de décisions à plusieurs niveaux. En partant d'un premier point du contour, il est possible de traiter le problème de minimisation qui, pour chaque ensemble fini d'étapes (*i*0, *i*1, …, *i*n-1), prend une décision parmi un ensemble fini de solutions possibles [44].

Le terme d'énergie interne est composé d'un terme du premier degré et d'un autre du second degré. Après la discrétisation, cette énergie interne met en jeu un élément du contour, son prédécesseur et son successeur.

On se ramène donc à un problème d'optimisation d'une fonction numérique de plusieurs variables. Les variables seront ici les positions des différents points du *snake.*

Chaque itération donne un contour optimal. La convergence de la minimisation de l'énergie est garantie, mais la complexité est élevée. Pour un voisinage de taille *m*, et un contour de *n* points, la complexité est O (*nm3*), la taille de la mémoire nécessaire est O (nm^2). La procédure est, par contre, parallélisable. Cette implémentation, d'autre part, autorise l'introduction de différentes contraintes comme l'énergie ballon ou d'autres énergies.

2.4.4.4 Algorithme Greedy :

La méthode du *Greedy* consiste à faire évoluer le contour actif en minimisant la fonctionnelle d'énergie : pour chaque point du *snake*, on choisit un nombre de voisins pour lesquels on va calculer l'énergie, le point se déplacera alors sur le voisin qui possède l'énergie la plus faible. On cherche donc l'ensemble *M* des points pour laquelle l'énergie est minimale. C'est donc un algorithme itératif qui déplace un point unique pour constituer un nouveau contour actif à chaque itération. Tous les points sont traités successivement lors de chaque itération [51].

L'utilisation de l'algorithme *Greedy* pour minimiser l'énergie d'un contour actif, comme l'ont proposé WILLIAMS et SHAH dans [45], est devenue une alternative assez fréquente à l'approche variationnelle.

WILLIAMS et SHAH, de manière classique et comme KASS et AMINI [40], discrétisent l'expression :

$$\boldsymbol{E_{tot}} = \int_0^1 \boldsymbol{E_{ext}}(\boldsymbol{v}(\boldsymbol{s})) + \frac{1}{2}\left(\boldsymbol{\alpha}(\boldsymbol{s})\left|\boldsymbol{v}'(\boldsymbol{s})\right|^2 + \boldsymbol{\beta}(\boldsymbol{s})\left|\boldsymbol{v}''(\boldsymbol{s})\right|^2\right) \text{ ds} \qquad \textbf{(2.15)}$$

Par différences finies et utilisent pour la continuité :

$$\left\|\frac{d\boldsymbol{v_i}}{\boldsymbol{ds}}\right\|^2 = \|\boldsymbol{v_i} - \boldsymbol{v_{i-1}}\|^2 = (\boldsymbol{x_i} - \boldsymbol{x_{i-1}})^2 + (\boldsymbol{y_i} - \boldsymbol{y_{i-1}})^2 \qquad \textbf{(2.16)}$$

Et pour la courbure :

$$\left\|\frac{d^2 v_i}{ds^2}\right\|^2 = \|\boldsymbol{v_{i-1}} - 2\boldsymbol{v_i} + \boldsymbol{v_{i+1}}\|^2 = (\boldsymbol{x_{i-1}} - 2\boldsymbol{x_i} + \boldsymbol{x_{i+1}})^2 + (\boldsymbol{y_{i-1}} - 2\boldsymbol{y_i} + \boldsymbol{y_{i+1}})^2 \qquad \textbf{(2.17)}$$

Il est à remarquer que ces deux expressions supposent deux hypothèses. La première est que les points soient placés, le long de la courbe, à une distance unité les uns des autres. Si les points sont équi-répartis, alors l'équation II.16 doit être divisée par d^2 et l'équation II.17 par d^4. Dans le cas contraire, la continuité est incorrecte d'un facteur di^2, où *di* correspond à la distance entre les points *i* et *i* –1. Cela impliquera une valeur supérieure de la continuité, de même pour la courbure. La seconde hypothèse suppose que le paramètre soit une longueur d'arc, pour que la courbure soit donnée par $||v_{ss}||^2$ [40].

L'algorithme de *Greedy*, comme la programmation dynamique autorise l'introduction d'autres contraintes. Il est plus rapide que la méthode précédente puisqu'il est, à chaque itération en O (*nm*) au lieu de O (nm^3) pour un contour comportant *n* points et un voisinage de taille *m* [40].

L'expression à minimiser, sans ajout de contraintes particulières, est :

$$E_{tot} = \int \left(\alpha(s)E_{cont} + \beta(s)E_{cour} + \gamma(s)E_{image}\right) \quad (2.18)$$

L'algorithme est itératif comme celui de Kass et d'Amini. On examine, à chaque itération, le voisinage de chaque point. On choisit le point du voisinage qui donne la plus faible valeur de l'énergie totale.

Après discrétisation, la minimisation de la distance entre les points implique une rétraction naturelle du contour. WILLIAMS et SHAH dans [45] proposent une variante qui évite cette rétraction trop importante. Ils utilisent la différence de distance entre les deux points $\|v_i - v_{i-1}\|$, par rapport à la distance moyenne des points du contour $\bar{d}$.

$$E_{cont} = \bar{d} - \|v_i - v_{i-1}\| \quad (2.19)$$

Les différents membres de l'expression sont normalisés avant minimisation. Les grandeurs utilisées sont le maximum, le maximum moins le minimum et le maximum moins le minimum avec saturation. L'énergie de continuité et l'énergie de courbure sont normalisées par division par la valeur du maximum dans le voisinage, donnant ainsi une valeur appartenant à l'intervalle [0,1]. La valeur de l'énergie externe ou d'image est normalisée, sur le voisinage, par (*min* - *val*)/(*max* - *min*). Afin d'éviter de trop grandes variations dans des zones relativement homogènes, c'est-à-dire avec une faible variation du gradient, si (*max* - *min*) < 5 alors la *min* sera mis à cinq. A chaque itération, une étape de calcul de courbure est effectuée. Lorsque la courbure devient supérieure à un seuil alors le coefficient β est mis à zéro, autorisant ainsi la présence de coins [40].

Une variante, permettant de rendre l'algorithme encore plus rapide, est proposée par LAM et YAN [52]. Il s'agit, par exemple, pour un voisinage de 3x3 pixels sur les 8 voisins, de n'en examiner que quatre. Si l'un de ces quatre améliore l'énergie totale, alors il n'est pas nécessaire d'aller plus loin. Sinon, les quatre qui restent sont examinés. Cela augmente le nombre d'itérations pour atteindre la convergence mais décroît le temps de calcul de chaque itération.

2.4.5 Contours actifs géométriques (Level Set) :

Dans le modèle d'origine des contours actifs, la courbe est représentée par une liste de points et l'évolution de la courbe par la variation des coordonnées de ces points. Caselles dans [53] introduit un modèle géométrique, puis dans [54] la notion de contour géodésique [55]. Malladi dans [56] implémente une évolution de la courbe basée sur un schéma de « level set » qui permet de gérer automatiquement les changements de topologie [57].

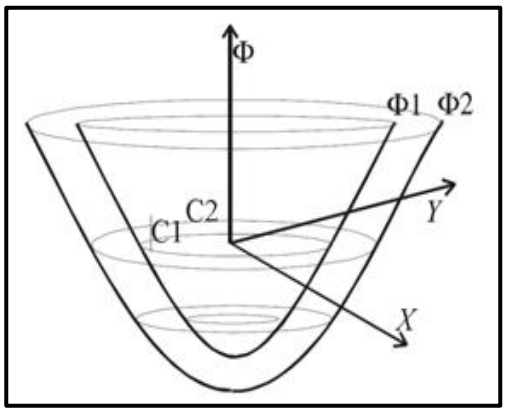


Figure 2.13: Formulation des « Level Sets ».

En rapport avec la figure II.13, pour une courbe fermée *c* dans le plan d'équation Φ=0, on peut construire une fonction Φ(x,y) telle que Φ(x,y) < 0 à l'intérieur de *c* , Φ(x,y) > 0 à l'extérieur de *c* et Φ(x,y)= 0 sur *c* .

La courbe *c* est appelée « level set » de Φ.

Si, on suppose, dans le plan Φ = 0, qu'une courbe évolue de c_1 a c_2 avec une certaine fonction vitesse, au lieu de faire évoluer la courbe, il est possible d'obtenir le même résultat en faisant évoluer une fonction Φ correspondante :

Faire Φ =Φ_1 pour qui « Level set » est c_1.

Faire décroître g.

2.5 Conclusion :

Dans ce chapitre nous avons présenté la méthode utilisée dans notre travail, qui est basée sur la segmentation d'images par contours actifs.

Le contour actif classique ou snake résulte d'une énergie qui caractérise les paramètres de la courbe et les contraintes liées à l'image. Le minimum est atteint lorsque le contour déformable se trouve sur les parois (frontières) de l'objet à détecter et vérifie les contraintes de régularité géométrique. Cependant ce modèle classique de contours actifs présente certains inconvénients ; notamment, nous citons :

-l'initialisation doit être près de l'objet à détecter : le contour initial doit être le plus proche possible de l'objet à détecter au risque de ne pas être attirer par l'objet.

-plusieurs objets ne peuvent être détectés simultanément.

-difficile à détecter les coins : notamment dans les concavités où les points du snake ne peuvent s'aventurer sans pénaliser leur énergie interne en augmentant leur courbure.

-problème de choix des paramètres : les paramètres d'élasticité et de courbure adaptés à une forme particulière ne sont donc pas utilisables sur une autre forme.

-sensible au bruit : si l'image est bruitée, il a de fortes chances de s'arrêter sur des contours parasites.

- De plus, ce type de contours a des difficultés de gérer automatiquement le changement de topologie de la courbe en cours d'évolution.

Le prochain chapitre sera consacré à l'implémentation de notre approche développée, pour la détection et le suivi des zones navigables en environnement d'extérieur structuré de type route ; aux tests et résultats de cette approche. Nous aborderons l'aspect pratique de notre travail, nous verrons l'algorithme implémenté, les applications développées et les résultats obtenues.

***Chapitre 3*: Implémentation**

Tests & Résultats

Conclusion Générale

Conclusion générale

Le travail présenté dans ce mémoire rentre dans le cadre de l'utilisation de la vision monoculaire appliquée à la navigation d'un robot mobile, de type voiture, sans intervention humaine dans le but d'avoir un véhicule autonome pour le transport urbain capable de transporter des passagers en toute sécurité.

Notre contribution consiste à utiliser des fonctionnalités visuelles dans le système embarqué à bord du robot mobile Robucar pour la détection des zones navigables. L'environnement choisi est un environnement d'extérieur structuré de type route, où nous avons utilisé une méthode de détection et suivi des régions navigables en exploitant uniquement une caméra noir et blanc.

L'approche que nous avons développée est basée sur la segmentation d'images. La détection des zones navigables est réalisée en utilisant la méthode des contours actifs (Snakes) adaptée pour extraire les contours. Nous avons commencé par l'acquisition des images en niveaux de gris jusqu'à l'extraction des contours des régions associées à ces zones. Cette approche garantit la sûreté, permettant au robot mobile Robucar de réagir et planifier ses mouvements en temps réel tout en évitant les collisions.

Nous avons rappelé, dans ce mémoire, des notions essentielles sur les robots mobiles autonomes. Par la suite, nous avons expliqué leurs aspects matériels et logiciels. Pour ensuite donner un aperçu général sur la perception en robotique mobile en citant les différents capteurs utilisés. En second lieu, nous avons présenté les différentes stratégies de navigation tout en détaillant les architectures de contrôle permettant à un robot mobile de réaliser les taches de navigation à savoir : 1) navigation en environnement d'intérieur avec deux approches qui sont : a) la navigation par carte métrique basée sur la localisation du robot, b) la navigation fondée sur une carte topologique ; 2) la navigation en environnement d'extérieur : a) la navigation non structurée (ou semi-structurée) concerne les chemins de terre des fermes, des terrains

accidentés, etc. et b) la navigation structurée basée sur le suivi de routes ou de chemins en terrain plat. Et enfin, nous avons parlé de notre approche de suivi concerne la détection et le suivi des zones navigables en temps réel en environnement d'extérieur structuré.

Par la suite, nous sommes intéressés aux contours actifs, où nous avons présenté les méthodes les plus connues dans la littérature à savoir: les variationnelles, par programmation dynamique et par algorithme de Greedy. Il s'agit dans tous les cas de minimisation d'une somme d'énergie diverses. Pour notre cas et après l'utilisation de la fonction cvSnake de la bibliothèque OpenCV, nous avons implémenté la méthode des snakes par l'algorithme de Greedy, pour sa rapidité, sa simplicité et sa facilité de mettre en œuvre. Cependant, elle présente des limitations parmi lesquelles : l'initialisation du contour qui doit être faite le plus près possible de l'objet à segmenter.

Les Snakes présentes d'autres limitations qui sont associées aux difficultés de convergence lorsque l'objet cible est constitué par des régions concaves. Et pour pallier à ces inconvénients nous exploitants la réinitialisation (réchantillonnage) périodique du processus d'extraction des zones navigables.

Et enfin, nous avons testé notre approche développée en simulation sur le Robucar. Une approche fondée sur la détection de contours a été implémenté pour extraire la forme des chemins où va se déplacer le robot mobile. Les résultats de la détection des chemins sont encourageants malgré les formes très variables des régions et le changement d'illuminations.

Parmi les difficultés rencontrées nous citons :

Les changements continus d'illuminations qui ont posé plusieurs problèmes à la technique de segmentation choisie ; l'influence des paramètres sur la détection des régions à fort contraste.

Les problèmes liés à l'aspect expérimental. L'environnement étant très complexe, les occultations, le changement de luminosité, la grande variation de texture.

Conclusion générale

Les problèmes imposés par le choix des paramètres qui s'adaptent avec l'image à segmenter.

Pour terminer, nous citons quelques perspectives concernant ce travail :

- La méthode proposée a trop de paramètres à régler ; donc il faut améliorer un mécanisme qui permet de contrôler ces paramètres automatiquement.
- Implémenter une autre méthode efficace et plus rapide qui converge dans les coins et les concavités comme le GVF.
- Réaliser un traitement spécifique ou prétraitement afin de reconnaitre et corriger les difficultés posées par l'analyse des chemins comportant de forts contrastes entre régions ombragées et ensoleillées.
- Doter le Robucar d'une caméra couleur pour implémenter des algorithmes de segmentation exploitant la couleur.
- Utiliser de la stéréovision (vision 3D) au lieu de la vision monoculaire dont le but d'améliorer l'extraction des régions navigables et estimer les positions des obstacles inattendus.
- Implémenter d'autres méthodes de suivi tels que : le suivi des objets, suivi d'autres véhicules en mouvement.

Annexe A : Robucar

A.1 Description du Robucar :

Le Robucar est un véhicule électrique construit sur la base d'un composé tubulaire des petites voitures. Il est utilisé comme plateforme expérimentale au sein de la division productique et robotique du CDTA (Figure A.1).

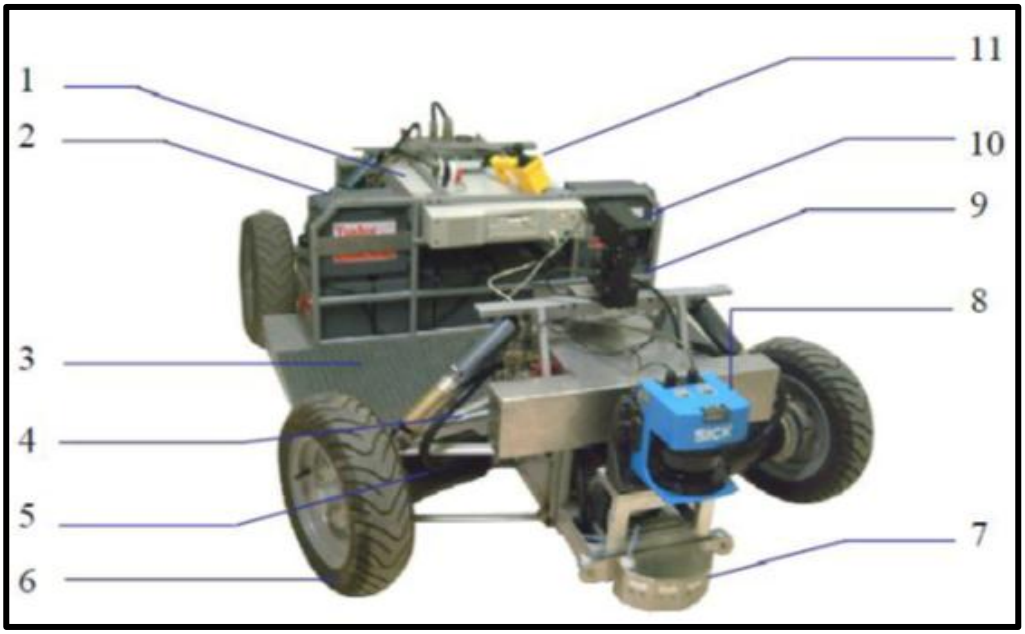


Figure A.1 : *Le Robot Mobile Robucar*.

1 : PC embarqué.

2 : 8 batteries.

3 : Châssis.

4 : Vérin électrique de direction.

5 : Un moteur de traction par roue.

6 : Roue motrice et directrice (avec un frein à commande électrique).

7 : Ceinture de capteurs à ultrasons (deux ceinture de 4 capteurs, avant et arrière).

8 : Capteur laser (SICK LMS).

9 : Ecran tactile.

10 : Caméra CCD (N/B).

11 : joystick pour la conduite manuelle sécurisée.

A.2 Caractéristiques techniques du Robucar :

Le robot mobile Robucar se déplace grâce à ses quatre roues motrices et directrices réparties en deux trains ; avant et arrière. Ces roues sont reliées à quatre moteurs qui assurent la traction du robot et deux vérins qui assurent la rotation des roues avant et arrière. Les caractéristiques générales du robot mobile Robucar sont les suivantes :

-longueur totale 1836 mm
-largeur totale 1306 mm
-hauteur : 616 mm
-poids totale : 310 kg
-Motorisation : 4 moteurs électriques de 1200 Watts.
-4 roues motrices et directrices.
-Vitesse maximale : 18 km/h (5 m/s).
-autonomie : 2 heures d'utilisation continue.
-capacité d'accueil : 2 personnes avec baguages.
-conduite : automatique ou manuelle.

Le Robucar est constitué d'un ordinateur embarqué et de deux cartes RSMPC555 équipées d'un microcontrôleur MOTOROLA MPC555. Ces cartes contrôlent respectivement les moteurs de traction avant et arrière ainsi que ceux de direction. L'ensemble PC embarqué et cartes MPC555 communique via un bus de terrain contrôlé area network (CAN).

Le robot est muni d'un capteur laser, d'une caméra CCD N/B placés sur la face avant du robot, de huit capteurs à ultrasons ; quatre placés à l'avant et quatre en arrière. Il comporte aussi des encodeurs (liés aux quatre moteurs de traction et aux deux vérins) qui servent à mesurer le déplacement et l'orientation effectués par le robot. Les données provenant de ces capteurs sont présentées de façon à donner la vitesse linéaire et l'angle de braquage du robot à partir desquelles peut être calculé sa position.

A.2.1 Environnement logiciel :

Le Robucar est piloté soit manuellement par un joystick, ou bien automatiquement. Pour contrôler le Robucar, deux variables sont exploitées à savoir sa vitesse instantanée ainsi que son angle de braquage. Ces deux variables sont envoyées vers une application en temps réel développée, sous un logiciel appelé SYNDEX v5.1 (SYNchronized Distribued EXecutive) [58].

A.2.2 Logiciel SYNDEX :

SYNDEX est un logiciel de CAO niveau système, fondé sur la méthodologie " Adéquation Algorithme Architecture " (AAA), pour le prototypage rapide et pour l'optimisation d'applications distribuées temps réel embarquées [58].

Il est réalisé et développé par l'unité de recherche INRIA [2] de Toulouse en France. Il est exploité par des machines multi composants (réseau de processeurs et circuits intégrés spécialisés). C'est aussi un logiciel graphique doté d'une interface interactive pour saisir les graphes représentant les algorithmes et les calculateurs multi composants [46].

Algorithme de Greedy :

Début

Répéter

Pour chaque point ***i***_courant de snake faire

Pour tous les points du voisinage de ***i***_courant faire

Calcul de l'énergie de ***i***_courant (E_courant) ;

FinPour

Pour chaque point du voisinage ***V*** de ***i***_courant faire

Calcul l'énergie de ***V*** (E_vi) ;

Normaliser l'énergie de ***V*** ;

FinPour

Choisir la plus petite Energie ***E_vmin*** parmi toutes les énergies voisines ;

Si (*E_vmin* < *E_courant)* alors :

Déplacer ***i***_courant vers le point qui a la plus petite énergie ***E_vmin*** ;

FinSi

FinPour

Jusqu'à atteindre l'énergie minimale de la courbe ou atteindre le nombre max d'itération ;

Fin

Bibliographies

Référence Bibliographies

[1] C.Drocourt, ''*Localisation et modélisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels*'', Thèse de Doctorat, Université de Technologie de Compiègne, Compiègne, France, 2002.

[2] T.Fraichard, ''*Cybercar : l'alternative à la voiture particulière*'', Papier proposé à la navigation, INRIA & Gravir-CNRS, septembre 2006.

[3] H.Bensakhri et M. Djerboua, ''*Détection des bords de la chaussée par vision monoculaire pour un robot mobile de type voiture*'', Projet de Fin d'étude, Université Saad Dahleb, Blida, Algérie, 2009.

[4] O.Boudar, "*Contribution au développement d'un système de détection des bords de la chaussée par vision monoculaire pour un robot mobile autonome* ", Projet de fin d'étude, Université des Sciences et de la Technologie Houari Boumediene, Alger, Algérie, 2010.

[5] M.KASS, A.WITKIN & D.TERZOPOULOS, ''*Snakes : Active Contour Models*'', International Journal of Computer Vision – IJVC, vol.1, no.X, 1988, pp.321-331.

[6] A.M.Hernandez ''*Vision dynamique pour la navigation d'un robot mobile*'', Thèse de Doctorat, Institut National Polytechnique de Toulouse, Toulouse, France, 2004.

[7] J.Pichon, ''*Guidage visuel d'un robot mobile autonome d'inspiration bionique*'', Thèse de Doctorat, Institut Nationale Poly technique de Grenoble, Grenoble, France, 1991.

[8] A.Lampe, ''*Méthodologie d'évaluation du degré d'autonomie d'un robot mobile terrestre*'', Thèse de Doctorat, Institut National Polytechnique de Toulouse, France.

[9] P.Coiffet, ''*La robotique – Principes et applications–Robots*'', Hermes, Traité des Nouvelles Technologies, série Robotique, 1986.

[10] S.Bouraine, ''*Localisation du robot mobile Robucar basée sur la méthode des grilles d'occupation et le filtre de Kalman étendu*'', rapport de recherche, CDTA, Centre de Développement des Technologies Avancées, Alger, Algérie, 2009.

[11] S.Gentil, ''*Intelligence artificielle appliquée à l'automatique, technique de l'ingénieur*'', traité mesure et contrôle, R 7251, 1996.

[12] A.Akli, ''*Elaboration d'une stratégie de coordination de mouvement pour un manipulateur mobile redondant*'', Projet de Fin d'étude, Université des Sciences et de la Technologie Houari Boumediene, Alger, Algérie, 2008.

[13] B.Bayle, ''*Support de cours (Robotique mobile)*''. Ecole Nationale Supérieure de Physique de Strasbourg, Université de Louis Pasteur, Strasbourg, France, 2008-2009.

[14] D.Filliat, "*Support de cours (Robotique mobile)*", Ecole Nationale Supérieure des Techniques Avancées (ENSTA) ParisTech, Paris, France, 5 octobre 2011.

[15] P.Aknin, *"Capteurs et traitement du signal dans les transports guidés*", Editions Hermés/Lavoisier, 2002.

[16] R.Bensalem et A.Cherifi, ''*Approches de navigation collective basée sur l'apprentissage par renforcement d'un groupe de robots mobiles autonomes* », Projet de fin d'étude, 2004.

[17] T.Duval, "*Robotique Mobile, 68hc11 Et Os Dédié*", Edition DUNOD, 2003.

[18] H.Maitre, ''*La détection des contours dans les images*'', Rapport d'activité, 2002.

[19] K.Abdelaziz et O.Said, ''*Etude de la manœuvrabilité d'un robot mobile autonome non holonome évaluant dans un espace restreint*'', Projet de fin d'études, Institut Nationale de formation en Informatique, Alger, Algérie, 1997.

[20] O.Hachour, ''*Contribution à la navigation de robots mobiles autonomes : Implémentation de la stratégie sur un circuit FPGA*'', Thèse de Magister, 2001.

[21] C.B.Madsen. & C.S.Andersen, "*Optimal landmark selection for triangulation of robot position*", in Robotics and Autonomous Systems, vol. 23, No. 4, pp. 277-292, 1998.

[22] Hayet, J.B., ''*Contribution à la navigation d'un robot mobile sur des amers visuels textures dans un environnement structuré*'', Thèse de doctorat, Université Paul Sabatier, Toulouse, France 2003.

[23] X.Clady, F.Collange, F.Jurie, & P.Martinet, "*Object Tracking with a Pan Tilt Zoom Camera Application to Car Driving Assistance*", in Proc. of *ICRA 2001*, pp. 1653-1658, 2001.

[24] R.Chatila, & S. Lacroix, "*Adaptive navigation for autonomous mobile robots*", in Proc of International Symposium on Robotics Research (ISRR'95), Munich, October 21-24, 1995.

[25] C.Avina, ''*Navigation visuelle d'un robot mobile dans un environnement d'extérieur semi structuré* », Thèse de doctorat de l'institut national polytechnique de Toulouse, France, 2005.

[26] R.Chapuis, A.Potelle, J.L.Brame & F.Chausse, ''*Real-Time Vehicle Trajectory Supervision on the Highway*'', International Journal of Robotics Research, vol. 14, no. 6, pages 531-542, Décembre 1995.

[27] C.Rosenberg, M.Hebert & S.Thrun, ''*Image Color Constancy Using KL-Divergence*'', In Eighth IEEE International Conference on Computer Vision, volume 1, pages 239-246, Vancouver, Canada, July 7-14 2001.

[28] M.GONZALES, R.C.WINTZ, ''*Digital Image Processing*'', Addison wesley.1977.

[29] « Cours vision artificielle », Master Systèmes de Vision et Robotique, Département d'Electronique, Université Saad Dahlab Blida, Blida, Algérie, 2010-2011.

[30] A.Boucher, ''*Cours de vision par ordinateur*'', Institut de la francophonie pour l'informatique (IFI).

[31] T.Alendre & N.Carayon & S.Soissons, ''*Egalisation d'histogramme*'', projet télécom, Paris 2006.

[32] J.P.Cocquerez et Philip, ''*Analyse d'image : Filtrage et segmentation*''. Edition Masson, 1995.

[33] N.Rezzoug, ''*Segmentation d'images par contours actifs floues'',* Projet de Fin d'étude, Université Saad Dahleb, Blida, Algérie, *2010.*

[34] H.M.Paindavoine, ''*Traitement des images en temps réel''*, technique de l'ingénieur, traité mesures de contrôle, R6720, 2000.

[35] D.Marr, ''*Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*'', Ed.W.H.Freeman and Company,New York, USA,ISBN:0-7167-1284-9, P 397, 1982.

[36] J.Canny, "*A Computational Approach to Edge Detection*''. IEEE. Transaction on PAMI, P 679-698, 1986, 8(6).

[37] A.Belaid, ''*Reconnaissance des formes*''. Edition InterEditions, 1991.

[38] J-J.ROUSSELLE, ''*Les contours actifs, une méthode de segmentation, application à l'imagerie médicale*'', Thèse de Doctorat, Université de Tours, France 9 Juillet 2003.

[39] M-O.BERGER, ''*Les contours actifs : modélisation, comportement et convergence*'', Thèse de doctorat de l'institut polytechnique de Lorraine, Février 1991.

[40] C. XU. ''*Deformable Models with Application to Human Cerebral Cortex Reconstruction from Magnetic Resonance Image*'' PhD Johns Hopkins University, Baltimore, Maryland, USA, p.135, Avril 2000. Disponible le 18/03/2000 sur : *http://iacl.ece.jhu.edu/~chenyang/research/pubs.html*

[41] C.XU & J.L.PRINCE, ''*Snakes, shapes and gradient vector flow*''. IEEE Trans. On image processing, vol. 7, n° 3. 1998. Disponible sur : *http://iacl.ece.jhu.edu/~chenyang/*

Référence Bibliographies

[42] C.XU, J.L.PRINCE, ''*Generalized Gradient Vector Flow External Forces for Actives Contours*'', Signal processing, Vol, 71 (2), Pages 131-139, Décembre 1998. Disponible le 09-02-2002, sur : *http://w.w.wiacl.ece.jhu.edu/projects/gvf/*

[43] L.D.COHEN, ''*On Active Contours Models and Balloons, Computer Vision, Graphics, and Image Processing: Image Understanding*'', vol. 53, n° 2. 1991. Disponible sur: *http://www.ceremade.dauphine.fr/~cohen/habilpub.html*

[44] A.AMINI, A.TERHANIS & E.WEYMOUTH, "*Using Dynamic Programming for Minimizing the Energy of Active Contours in the Presence of Hard Constraints*'', 1988.

[45] D.J.WILLIAMS & M.SHAH, ''*A Fast Algorithm for Active Contours and Curvature Estimation*'', CVIGP Computer Vision Graphics Image Process: Image Understanding, vol. 55, n° 1, p. 14-26, Janvier 1992.

[46] J.DENZEL & H.NIEMANN, "*Evaluating the Performance of the Avtive Contour Models for Real-time Object Tracking*'', 1995. Disponible sur: *http://www.informatik.unienlargen.de/litérature/english/bild/1995.hml*

[47] L.D.COHEN, ''*Etude de modèles de contours actifs et d'autres techniques de traitement d'images*'', Thèse de Doctorat Université Paris Sud, centre d'Orsay, France, 1990.

[48] L.D.COHEN & I.COHEN, ''*IEEE Transactions on Pattern Analysis and Machine Intelligence*'', PAMI, vol. 15, n° 11, Novembre 1993.

[49] A.COUTANT, ''*La méthode des contours actifs en traitement des images'',* Conservatoire National des Arts et Métiers, Paris, le 02 février 2005.

[50] L.D.COHEN, ''*On Active Contours Models*'', In Proceeding of NATO ASI active Perception and Robot vision, Maratea, Juillet 1989.

[51] J.Michot, ''*amélioration des paramètres des contours actifs par une technique issue de la théorie de l'apprentissage*'', 2002-2004

[52] K.-M.LAM & H.YAN, ''*Fast Greedy Algorithm for Active Contours*'', Electronics letters, Janvier 1994.

[53] V.Caselles, F.Catté, T.Coll & F.Dibos, '' *A Geodesic Model for Active Contours in Image Processing*'', Namerische Mathematics, N° 66, P 1-31, 1993.

[54] V.Caselles, R.Kimmel & G.Sapiro, ''*Geodesic Active Contours*'', Fifth International Conf. On Computer Vision (ICCV'95), Cambridge, MA, USA, P 694-699, Juin 1995. Disponible le 05-12-2002 sur: *http://w.w.w.iua.upf.es/~vcaselles/papers_v/ICCV_GAC.pdf.*

[55] V.CASELLES, R. KIMMEL, G. SAPIRO, ''Geodesic Active Contours'', International Journal of Computer Vision, vol. 1, n° 22, p 61-79, 1997.

[56] R.MALLADI, J.A.SETHIAN, B.C. VEMURI, B.C, '' *Shape Modeling with Front Propagation: A Level Set Approach*'', IEEE T.Patt, vol. 17(2), 1995, p. 158-175. Disponible le 28/12/2001 sur : http://citeseer.nj.nec.com/malladi95shape.html.

[57] J.A.Sethian, ''*Level Set Methods: Evolving Interfaces in Computer Vision Mechanics*'', Computer Vision and Materials Sciences, Cambridge University Press, 1996-1999.

[58] *www.syndex.org, SYNDEX, 2006.*

Dans ce présent chapitre nous montrons en détails l'application que nous avons développée ainsi que les résultats obtenus.

Nous commençons d'abords par présenter l'environnement de travail, les travaux précédents, les langages de programmation et la bibliothèque utilisée (OpenCV).

3.1 Objectif de notre travail :

Notre objectif principal est de permettre à un RMA de type voiture, Robucar, de détecter et de suivre les zones navigables sur son chemin, et ceci, en utilisant la vision monoculaire. Sachant que ce robot se déplace dans un milieu urbain structuré, avec présence de plusieurs objets mobiles et fixes tels que d'autres véhicules, piétons, etc.

Notre mission est de faire de telles sortes que ce robot mobile arrive à naviguer, sans collisions, tout en connaissant les zones navigables contenues parmi les obstacles de son environnement.

3.2 Les Moyens Utilisés :

3.2.1 Matériels :

Le matériel utilisé dans ce travail est composé d'un ordinateur portable de marque HP pour la simulation et le développement software, une webcam de bonne résolution, une caméra noir/blanc et le robot mobile Robucar.

Le Robucar est un véhicule électrique construit sur la base d'un composé tubulaire des petites voitures. Il est utilisé comme plateforme expérimentale au sein de la division productique et robotique du CDTA (voir annexe A).

Notre application a été testée en simulation sur un ordinateur ayant les caractéristiques suivantes:

- Micro-processeur: Intel Core i3 à vitesse d'horloge de 2.53Ghz.
- Capacité mémoire : Ram 4GO.
- Disque dur : Samsung 500GO.
- Webcam : HP TrueVision2Mpixels.

3.2.2 Langages de programmation :

Notre application a été développée sous Windows 7. Pour la réalisation et l'implémentation, nous avons utilisé en premier lieu le langage de programmation C^{++} Builder6 mais nous avons rencontré plusieurs problèmes, nous citerons par exemple la configuration de la dernière version de la librairie OpenCV. Pour cela, nous avons passé à un autre langage de programmation d'actualité, plus simple, très performant et plus praticable le Visual Studio $C^{\#}$ 2008.

3.2.2.1 C^{++} Builder6 :

Le C++ Builder6 est un environnement de développement proposé par la société Borland fondé sur le C++. Pour cet environnement de développement, Borland a repris les recettes développées avec succès pour ses produits phares orienté Pascal : Delphi ; notamment l'interface et les bibliothèques de composants.

Le C++ Builder6 est un outil RAD (Rapid Application Development), c'est-à-dire tourné vers le développement rapide d'applications sous Windows. Le C++ Builder6 permet de réaliser de façon très simple l'interface des applications et de relier aisément le code utilisateur aux événements Windows, quelle que soit leur origine (souris, clavier, système). Pour ce faire, le C++ Builder6 repose sur un ensemble très complet de composants (visuels et non visuels) prêts à l'emploi. Les caractéristiques de ces composants sont éditables directement dans une fenêtre spéciale intitulée inspecteur d'objets. Un clic de souris sur les composants permet d'associer du code à l'objet.

Le C++ Builder6 propose deux hiérarchies de composants :

- La bibliothèque VCL ou Visual Component Library dédiée à Windows.
- La bibliothèque CLX pour des composants multiplateformes pour Windows.

La figure 3.1 représente un exemple typique de l'interface de C++ Builder6 au cours d'une session de travail.

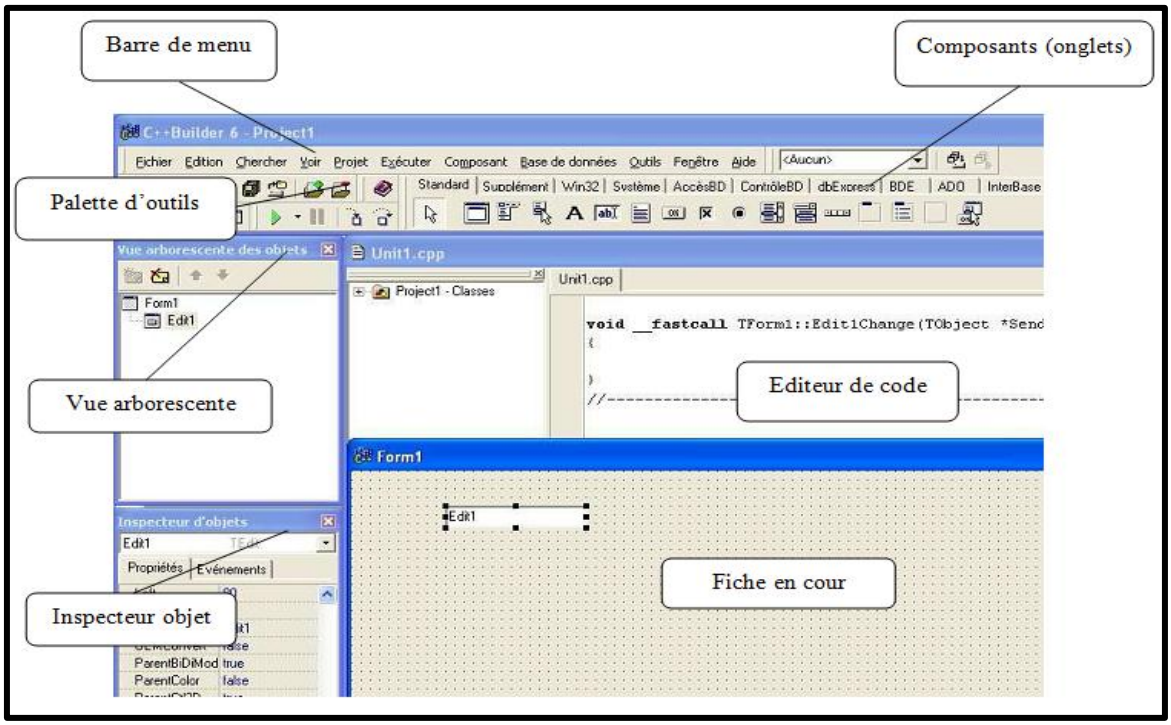


Figure 3.1 : L'interface de C++ Builder6.

On peut distinguer plusieurs zones:

- La barre de menu.
- La barre d'outils qui se décompose elle-même en deux parties :
  - La palette d'outils pour les opérations classiques.
  - La palette de composants rangés par catégories (onglets).
- Une fiche ou (Form) qui représente l'interface en cours de création.

3.2.2.2 Microsoft Visual Studio C#:

Microsoft Visual Studio C# est un langage orienté composant (objet) créé par Microsoft. C# joue un rôle essentiel dans l'architecture de Microsoft.

NET Framework, et certaines personnes ont comparé son rôle à celui joué par le langage C dans le développement de l'UNIX. La syntaxe de C# en est très proche de C^{++}.

La figure ci-dessous montre l'interface de Visual C# au cours de travail.

On peut distinguer les taches suivantes :

1. Le Toolbox pour choisir n'importe qu'elle instruction.

2. La barre de menu.

3. le fichier [Form1.cs] en mode "conception" (design).

4. le projet [WindowsFormApplication] dans la solution [WindowsForm] :

• [Program.cs] est la classe principale du projet.

• [Form1.cs] est le fichier source qui va gérer le comportement de la fenêtre [3].

• [Form1.Designer.cs] est le fichier source qui va encapsuler l'information sur les composants de la fenêtre [3].

5. Editeur des erreurs.

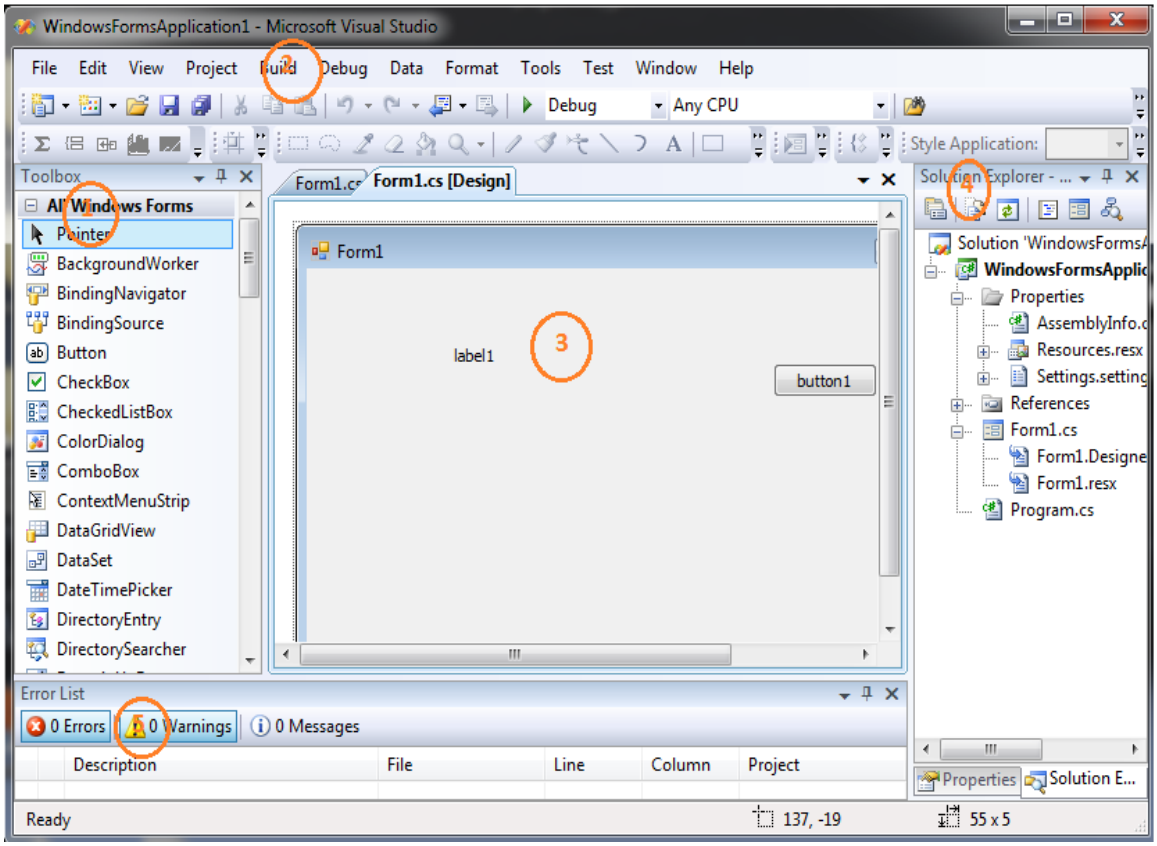


Figure 3.2: L'interface de Visual C#.

3.2.3 Bibliothèque OpenCV :

OpenCV (Open Source Computer Vision) est une bibliothèque optimisée, de fonctions dédiées à la programmation pour la vision en temps réel. Cette librairie publiée sous une licence BSD est gratuite pour un usage scolaire ou commercial. Son intérêt est de réaliser des traitements sans forcément connaître les algorithmes. La bibliothèque possède plus de 500 algorithmes optimisés. Elle est utilisée dans la vision en contexte robotique pour des applications simples.

La librairie OpenCV se présente de la manière suivante :

- **CV & CVAUX**: **Image processing and vision algorithms.**

-Gradient, contours, coins et contours actifs, etc.

-Morpho-math (érosion, dilatation, fermeture, etc.).

-Filtrages divers (lissage, rehaussement de contraste, suppression de fond, etc.).

-Conversion d'espace couleur (RGB, HSV, YCbCr, etc.).

-Étiquetage, manipulation de contours, Transformations diverses (Fourier, Hough...)

-Histogrammes.

-Analyse et suivi de mouvement.

- **CXCORE: Basic structure and algorithms & drawing functions.**

-Structures élémentaires: matrices, tableaux, listes, files, graphes, arbres, etc.

-Dessin de primitives géométriques :lignes, cercles, ellipses, arcs, polygone plein ou contours, textes (avec différentes fonts), etc.

- **HIGHGUI :**

-Manipulation des images et des séquences : lecture, écriture, sauvegardé, etc.

-Interface utilisateur.

-Fenêtre, entrées/sorties utilisateur.

3.3 Implémentation :

Après avoir exposé les différentes techniques retenues pour la réalisation de notre application, nous allons présenter l'environnement de travail, la structure et les fonctionnalités de notre application.

3.3.1 Environnement de Travail :

A. Implémentation sous C^{++} Builder6 :

La figure suivante illustre l'interface crée sous C^{++} Builder6:

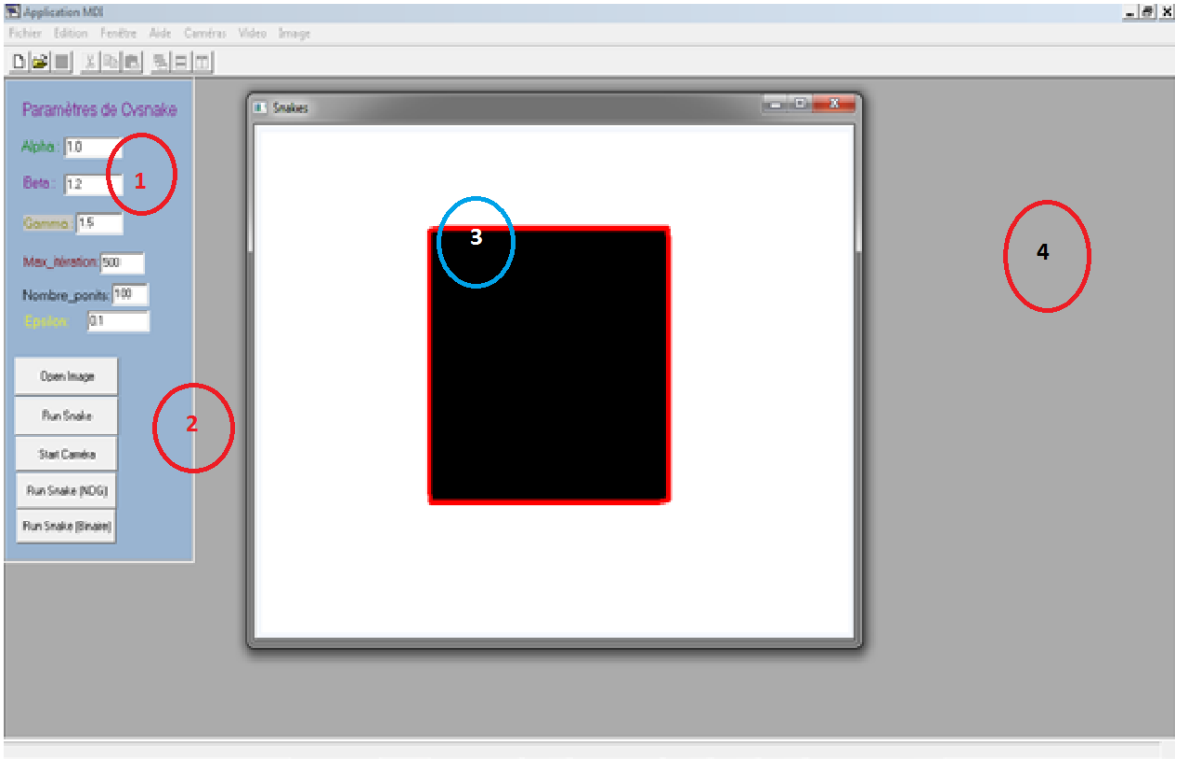


Figure 3.3: Interface utilisateur de C^{++} Builder6.

On peut distinguer facilement dans l'interface les menus suivants :

(1) Le panneau de configuration du cvSnake (α, β, ɣ), ainsi le nombre de points contenus dans le contour et le nombre d'itérations.

(2) Le panneau de configuration de la webcam, open image et exécuté le snake.

(3) L'image à segmenter et l'évolution du contour à l'intérieur de l'image.

(4) La Forme qui représente l'interface crée.

B. Implémentation sous Visual Studio C# :

La figure suivante montre un aperçu général de notre application développée:

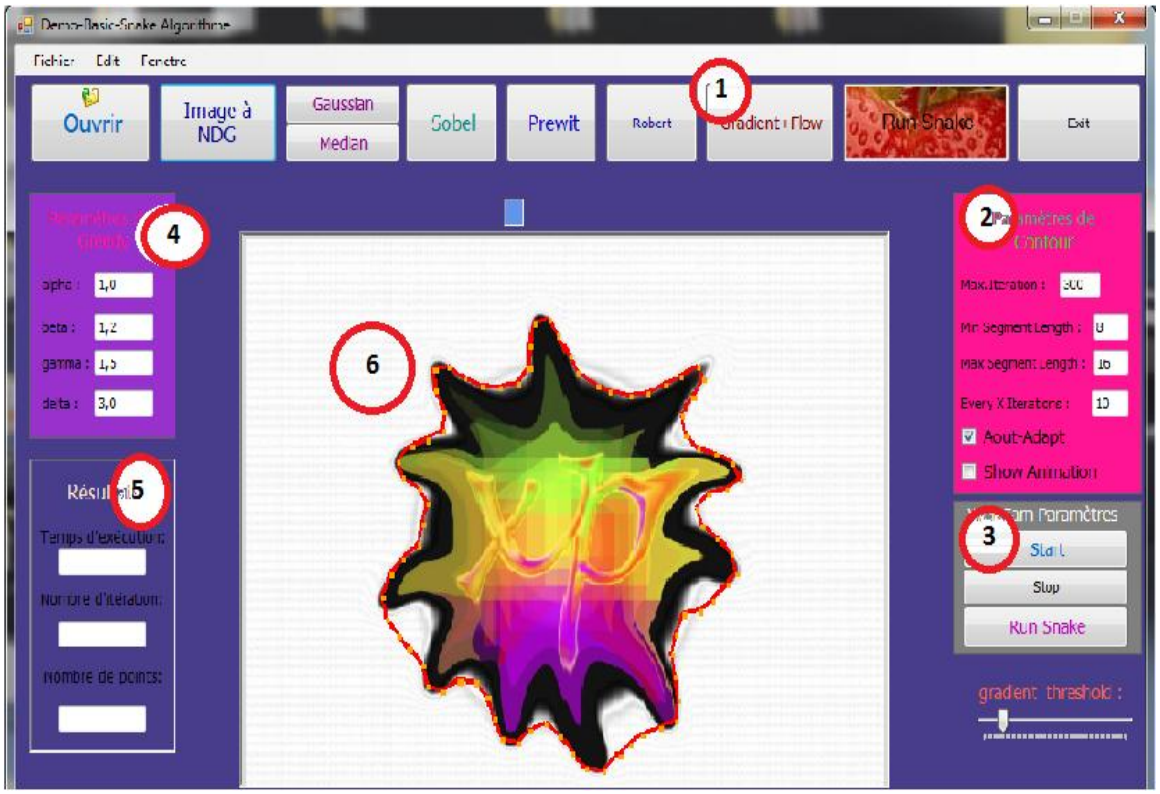


Figure 3.4: Interface utilisateur de Visual C# (notre application développée).

L'interface utilisateur est structurée de telle sorte à faciliter son utilisation, ainsi on peut distinguer facilement :

(1) La barre de menu, qui regroupe toutes les fonctionnalités de l'application.

(2) Le panneau de configuration du contour, qui permet de définir les paramètres de la méthode qu'on a utilisée.

(3) Le panneau de configuration de webcam.

(4) Le panneau de configuration du Greedy (α, β, ɣ, δ).

(5) Le résultat de la segmentation (le nombre de points de Snake et le nombre d'itération).

(6) L'image à segmenter et l'évolution du contour à l'intérieur de l'image.

3.3.2 Schéma général:

Avant d'introduire le principe de la méthode de segmentation développée, nous allons donner un schéma qui résume les différentes étapes de segmentation et qui représente l'architecture générale de notre application (voir la figure 3.5). Nous avons en entrée un flux d'images acquises par une caméra, ce dernier doit passer par une phase de prétraitement et cela a pour but d'éliminer les bruits et de faciliter l'extraction des différentes zones de l'image. Ensuite, l'utilisateur aura à choisir les paramètres qui donnent des bons résultats de segmentation.

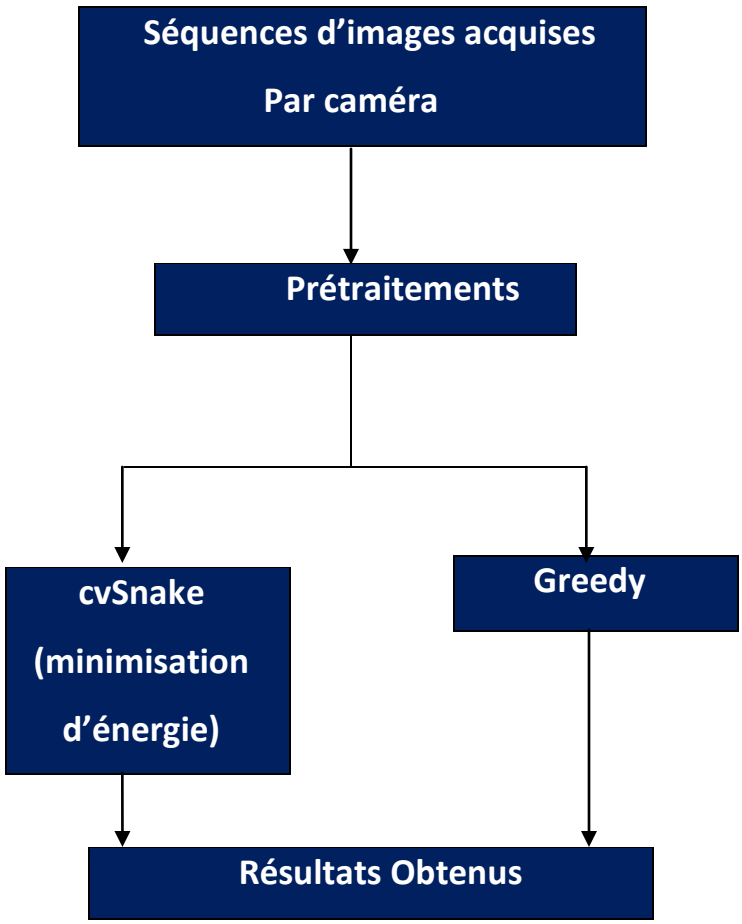


Figure 3.5: Schéma général de segmentation d'une image.

3.3.3 Fonctionnalité de l'interface:

3.3.3.1 Acquisitions des images :

Les images acquises sont issues d'une caméra noir et blanc embarquée dans le Robucar.

3.3.3.2 Prétraitements :

a. Filtre de Contraste :

On applique souvent un filtrage de contraste pour rendre les contours plus visibles. Cela est réalisé par un produit de convolution entre l'image et la matrice suivante comme montre l'algorithme 3.1:

```
Pour tout point I(i,j) de l'image I faire
    I(i,j)= -I(i-1,j) – I(i,j-1) + 5*I(i,j) – I(i,j+1) – I(i+1,j);
FinPour
```

ALGORITHME 3.1: Application d'un filtre de contraste.

b. Opération de Lissage :

Il existe plusieurs types de filtres permettant le lissage des images, ils permettent de réduire les bruits qui dégradent la qualité de l'image. On utilise pour cela soit des filtres gaussiens ou bien des filtres moyens.

b.1 Filtre gaussien :

L'image sera multipliée par un masque Gaussien à trois ou bien à cinq dimensions.

Pour un filtrage gaussien, l'opération de lissage s'effectue comme suit :

```
Pour tout point I(i,j) de l'image I faire
    I(i,j) = [ I(i-1,j-1)+ 2* I(i-1,j)+ I(i-1,j+1) + 2* I(i,j-1) + 4* I(i,j) + 2* I(i,j+1) +
            I(i+1,j-1) + 2*I(i+1,j) + I(i+1,j+1) ];
    I(i,j) = I(i,j)/16;
FinPour
```

ALGORITHME 3.2: Application d'un filtre Gaussien.

b.2 Filtre moyen:

De même que pour le filtrage gaussien, le filtrage moyen s'effectue par une multiplication de l'image par une matrice de dimension cinq ou trois.

Pour un filtrage moyen, l'opération de lissage s'effectue comme suit :

Pour *tout point* I(i,j) *de l'image* I **faire**

I(i,j) = [I(i-1,j-1) +I(i-1,j) + I(i-1,j+1) + I(i,j-1) +I(i,j) + I(i,j+1) +

I(i+1,j-1) +I(i+1,j) + I(i+1,j+1)];

I(i,j) = I(i,j)/9;

FinPour

ALGORITHME 3.3: Application d'un filtre Moyen

c. Opérateurs de Gradient :

Pour obtenir le gradient d'une image, on calcule la valeur de chaque pixel de l'image en fonction de ses voisins pondérés par une matrice de coefficient déterminés.

Ainsi, pour obtenir le gradient d'une image, on peut utiliser les méthodes de Roberts, de Prewitt ou bien de Sobel.

Pour notre cas, nous utilisons le filtre de Sobel.

Pour *tout point* I(i,j) *de l'image* I **faire**

∇I_X = I(i-1,j-1) + 2*I(i-1, j) + I(i-1,j+1) – [I(i+1,j-1) + 2*I(i+1,j) + I(i+1,j+1)];

∇I_y = I(i-1,j-1) + I(i+1,j-1) +2* I(i, j-1) – [I(i-1, j+1) + 2*I(i, j+1) + I(i+1,j+1)];

$\nabla I(i,j) = \sqrt{\nabla \mathrm{Ix}^2 + \nabla \mathrm{Iy}^2}$;

FinPour

ALGORITHME 3.4: Calcul le gradient de sobel.

3.4.3.3 Segmentation par l'algorithme de Greedy :

Comme nous l'avons déjà vu dans le deuxième chapitre, l'algorithme de Greedy (voir annexe B) est rapide et facile à implémenter ; son principe est d'examiner à chaque itération, le voisinage de chaque point en cherchant celui qui donne la plus faible valeur de l'énergie et cela pour déterminer la nouvelle position de ce point. Cela revient à minimiser

l'énergie totale qui est la somme de toutes les énergies des points. Le schéma suivant montre les différentes étapes de détection de contours par la méthode de Greedy.

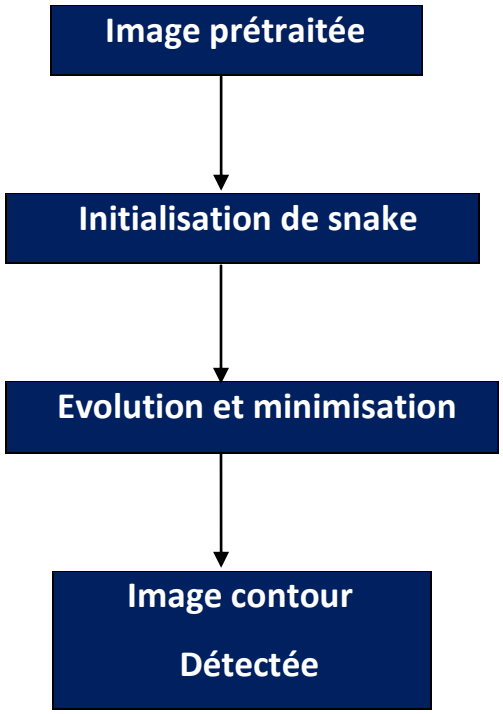


Figure 3.6: Schéma de détection de contours par la méthode de Greedy.

Chaque étape est détaillée dans ce qui suit :

a. Initialisation du contour:

On initialise notre contour actif à l'intérieur de l'objet ou de telle sorte qu'il englobe l'objet dont on veut détecter ses contours. Ce contour actif est composé d'un ensemble des points (snaxels), et il est initialisé sous forme d'un cercle (voir figure 3.7).

Pour cela, on doit initialiser seulement le centre du cercle et son rayon. Les coordonnées du snaxels sont calculées par la formule suivante :

$$x_i = x_c - r * \sin(2\pi i/N)$$

$$y_i = y_c - r * \cos(2\pi i/N) \qquad \text{pour } i=0..N-1$$

Où :

- x_i , y_i les coordonnées du point i du Snake.
- x_c , y_c les coordonnées du centre de cercle.
- r le rayon du cercle.
- N le nombre des points contenus dans la courbe (Snake).

Figure 3.7: Initialisation de la courbe dans l'image.

b. Evolution et minimisation :

La méthode de Greedy consiste à faire évoluer un contour initial en minimisant une fonctionnelle d'énergie. La minimisation de la courbe revient à minimiser l'énergie de chaque point de la courbe (Snake). Les points de la courbe se déplacent vers le voisinage qui minimise leur énergie. Pour maintenir l'homogénéité de la courbe, un échantillonnage est effectué à chaque itération. Des points peuvent être créés ou supprimés de façon à conserver une distance homogène entre les points voisins.

Le réchantillonnage garantie que la condition suivante est vérifiée :

$$D_{min} < || P_i - P_{i+1} || < D_{max}$$

Où : D_{min} et D_{max} deux bornes choisies par l'utilisateur.

Soit **d** la distance entre deux points quelconques de la courbe. Si les deux points sont très éloignés ($d > D_{max}$), alors on crée un nouveau point et s'ils sont trop près c.à.d. ($d < D_{min}$) alors il faut supprimer l'un des deux points (point_curant ou point_suivant).

- **Création d'un nouveau point :**

Cette opération consiste à ajouter un nouveau point entre deux points **pi** et **pi+1** si la distance **d** entre ces derniers est supérieure à Dmax. Son but est de garder un échantillonnage suffisant lorsque le contour se dilate. L'algorithme est le suivant :

```
Création d'un point (d>Dmax)
Pour chaque point du snake de position i faire
    Point_cur = snake[i];
    Point_next = snake[i+1];
    d = distance2D (Point_cur, Point_next) ;
    Si (d>Dmax) alors : créer un nouveau point entre Point_cur et Point_next ;
    FinSi
Finpour
```

ALGORITHME 3.5: Création d'un nouveau point.

- **Suppression d'un point :**

Son principe consiste à fusionner les deux points **pi** et **pi+1** en un seul en supprimant l'un des deux si la distance **d** est inférieure à Dmin. Le but de cette fusion est d'éviter les intersections entre les arêtes liant les sommets du contour lors de son évolution. L'algorithme de cette opération est le suivant:

```
Fusionner un point (d<dmin)
Pour chaque point du snake de position i faire
    Point_cur = snake[i];
    Point_next = snake[i+1];
    d = distance2D (Point_cur, Point_next);
    Si (d<Dmin) alors: supprimer le point suivant (Point_next);
    FinSi
FinPour
```

ALGORITHME 3.6: Suppression d'un point.

L'évolution du contour par l'algorithme de Greedy s'effectue de manière à minimiser une fonctionnelle d'énergie E_{tot} qui peut s'exprimer par la formule suivante :

$$E_{totale} = a*E_{continuité}(Pi) + b*E_{courbure}(Pi) + c*E_{ballon}(Pi) + d*E_{gradient}(Pi)$$

Les différentes énergies sont calculées comme suit :

- **Energie de continuité :**

Après la discrétisation de l'équation de l'énergie, la minimisation de la distance entre les points implique une rétraction naturelle du contour. Wiliams et Shah dans [45] propose une variante qui évite une rétraction trop importante**.** Ils utilisent la différence de distance entre les deux points, $|| P_i - P_{i-1} ||$ par rapport à la distance moyenne des points du contour. Cette énergie se calcule par rapport à la moyenne des distances entre les points du contour actif, telle que :

$$E_{Continuité}(P_i) = 0.5*abs\,[Dm - ((X_i - X_i\text{-}1)^2 - (Y_i - Y_{i-1})^2]$$

Pour minimiser cette expression, le point P_i doit se positionner à une distance égale à La distance moyenne (*Dm*) du point P_{i+1}.

La distance moyenne (*Dm*) se calcule par l'algorithme suivant:

```
Distance_myenne (Point_cur, Point_next)
Pour chaque point du snake de position i faire
    Dm=Dm + 0.5*[( Point_next.x – Point_cur.x)^2 + (Point_next.y – Point_cur.y)^2];
FinPour
Returner  Dm=Dm / Nbr de point du contour actif ;
```

ALGORITHME 3.7: Calcul la distance moyenne entre 2 points.

L'énergie de continuité est donnée par l'algorithme suivant:

```
Continuité (Point_cur, Point_next)
Début
Pour chaque point du snake de position i faire
    Pour chaque point de voisinage 3x3 faire
    D =distance (Point_cur, Point_next) ;
    FinPour
 Retourner  E_continuité = || D – Dm|| ;
 FinPour
 Fin
```

ALGORITHME 3.8: Calcul de l'énergie de continuité.

- **Energie de Courbure :**

L'énergie de courbure définit l'angle formé par trois points adjacents du contour actif. Plus le coefficient attribué à cette énergie sera fort, et plus le contour actif aura la forme connue pour être la forme la moins coûteuse en énergie : la ligne droite ou le cercle pour des objets fermés.

L'énergie de courbure est calculée par la relation suivante :

$$E_{Courbure}(Pi) = Sqrt((X_{i-1} - 2* X_i + X_{i+1})^2 + (Y_{i-1} - 2*Y_i + Y_{i+1})^2)$$

```
Courbure (Point_prev, Point_cur, Point_next)
Début
Pour chaque point du snake de position i faire
    Pour chaque point de voisinage 3x3  faire
    Retourner E_courbure =  sqrt [(Point_prev.x - 2* Point_cur.x + Point_next.x) ^2 +
                            (Point_prev.y - 2* Point_cur.y + Point_next.y)^2] ;
    FinPour
FinPour
Fin
```

ALGORITHME 3.9: Calcul de l'énergie de courbure.

- **Energie de gradient (externe) :**

Rappelons que le gradient de l'image est la première dérivée de la fonction d'intensité de cette dernière. Il est considéré comme l'élément de base pour mesurer les contours d'une image. Afin de trouver les changements brusques de la fonction d'intensité d'une image (les contours), on cherche les extrema de cette fonction. Malheureusement, il n'existe pas une forme linéaire pour cela, car on dispose de ces valeurs seulement à des points précis (pixels). L'idée est donc d'essayer de détecter les extrema en utilisant une estimation des dérivées de la fonction. Cela se fait par l'application d'un opérateur dans les deux directions x et y.

Voici l'algorithme de détection du gradient par l'opérateur de Sobel (il est détaillé dans l'algorithme 3.4).

Début

Pour chaque point de l'image I(i,j**) faire**

Calculer Grad_x(i,j) de Sobel ;

Calculer Grad_y(i,j) de Sobel ;

$Grad(_{i,j}) = sqrt(Grad\_x(_{i,j})^2 + Grad\_y(_{i,j})^2)$;

FinPour

Fin

ALGORITHME 3.10: Calcul du gradient par Sobel.

- **Energie de Ballon :**

L'énergie de Ballon est l'énergie qui décide du sens de propagation du contour actif. Un coefficient d'énergie de ballon positif va concentrer le snake, alors qu'un coefficient négatif va rendre le snake expansif.

L'énergie de ballon est calculée par l'algorithme qui suit :

Début

Pour chaque point du snake de position i **faire**

Pour chaque point de voisinage 3x3 **faire**

E_{ballon} = abs (sqrt ((Point_cur.x-X_c)^2 + (Point_cur.y-Y_c)^2)*sqrt ((Point_cur.x- Point_prev.x)^2 + (Point_cur.y- Point_prev.y)^2)) ;

FinPour

FinPour

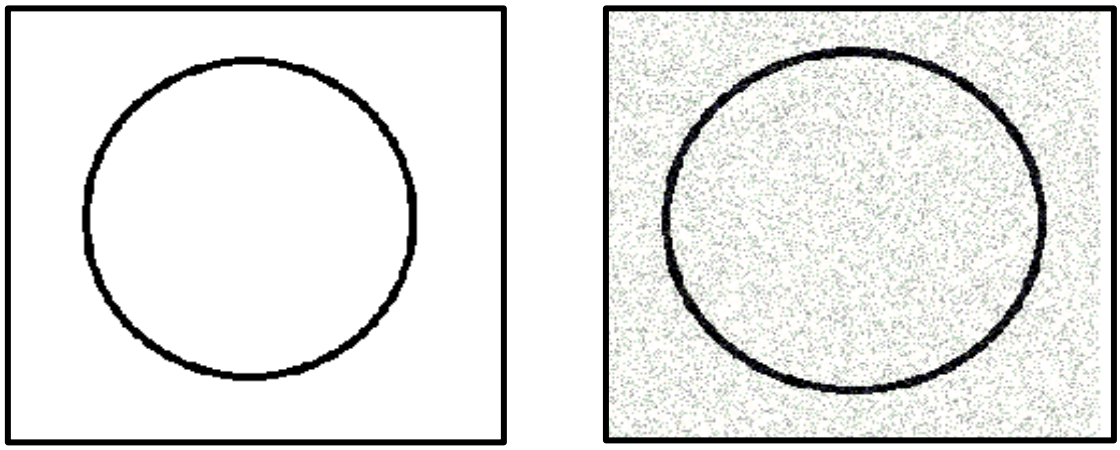
Fin

ALGORITHME 3.11: Calcul l'énergie de ballon.

3.4 Tests et Résultats :

Dans cette partie, nous présentons les tests que nous avons effectués sur les approches proposées, la détection de contour par l'algorithme de Greedy et par la fonction cvSnake existe en OpenCV ; où on jouant sur les paramètres des énergies.

- **Images de synthèses simples sans et avec présence de bruit :**



(a) (b)

Figure 3.8: Images synthèses avec objet simple.

a) Image sans bruit, b) Image avec bruit.

- **Images avec concavités:**

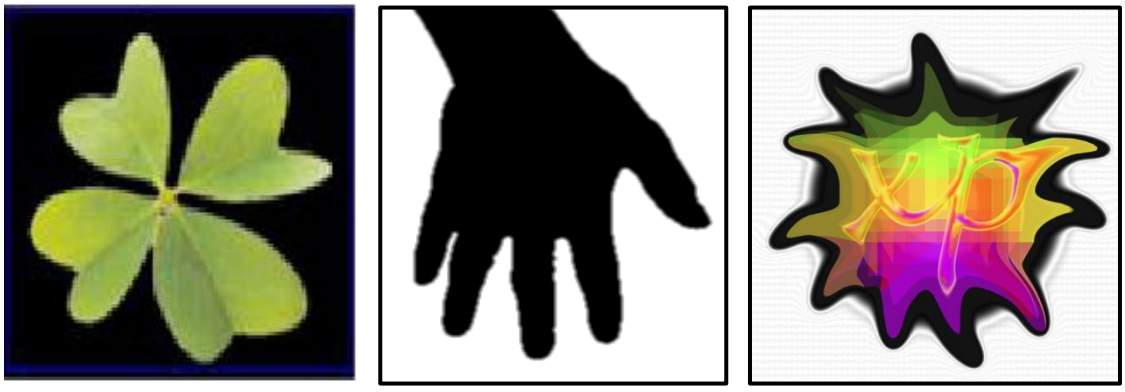


Figure 3.9: Images réelles avec concavités.

- **Images réelles :**

(a) (b)

Figure 3.10: Images réelles des chemins.

a) environnement semi-structuré, b) environnement structuré.

3.4.1 Tests sur l'influence des paramètres de CVSNAKE et de Greedy :

Dans ce qui suit nous allons montrer l'influence des paramètres sur les résultats de la segmentation, Nous commençons par les paramètres de la fonction cvSnakeImage de librairie OpenCV, ensuite nous passons aux paramètres de l'algorithme Greedy.

Nous avons choisi d'effectuer ces tests sur des images avec concavités pour voir clairement l'influence des coefficients pour détecter les contours.

3.4.1.1 Tests sur l'influence des paramètres de la fonction cvSnake :

En premier temps, nous avons utilisé la bibliothèque OpenCV sous l'environnement C^{++} Builder6 ; cette librairie contient une fonction connue sous '**cvSnakeImage',** qui se base sur une minimisation d'une somme d'énergie qui est représentée par l'énergie de continuité contrôlé par le coefficient α, l'énergie de courbure contrôlé par le coefficient β et l'énergie de gradient contrôlé par le coefficient ɣ**.**

Cette fonction contient plusieurs paramètres, et qui sont :

```
cvSnakeImage (IplImage*src, CvPoint* points, int length,
        float* alpha, float* beta, float* gamma,
        int coeff_usage, CvSize win, CvTermCriteria criteria,
        int gradient=1);
```

-src: image source qui on veut la segmenter doit être en NDG .

-points: points de contour actif (snake).

-length: Nombre de points dans le contour actif.

-alpha: coefficient qui contrôle l'énergie de continuité (interne).

-beta: coefficient qui contrôle l'énergie de courbure (interne).

-gamma: coefficient qui contrôle l'énergie de gradient (externe).

-coeff_usage: dépend des valeurs des paramètres précédentes, il prend 2 valeurs :

***CV VALUE :** utilisé si les valeurs de α, β et ɣ sont des nombres réels.

***CV ARRAY:** utilisé si les valeurs de α, β et ɣ sont des vecteurs (ensemble).

-win: la taille de voisinage de chaque point de contour actif.

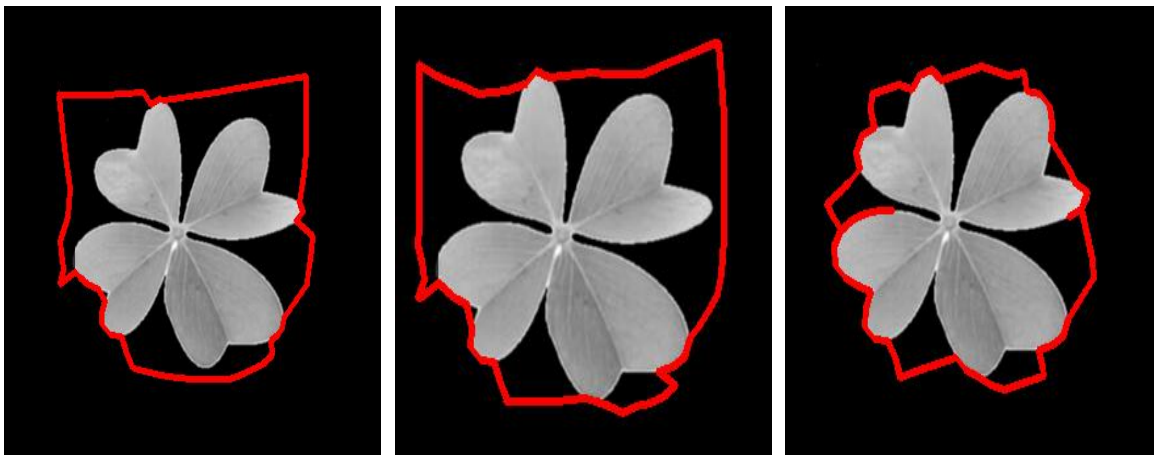
-criteria: Termination criteria.

-gradient: gradient de l'image à segmenter; prend les valeurs 0 ou 1.

Nous montrons, dans ce qui suit, l'influence des coefficients des énergies sur les résultats de la segmentation.

A. Tests sur le paramètre de continuité Alpha :

Le paramètre α permet de contrôler l'énergie de continuité. L'énergie de continuité fait partie des énergies dites internes au contour actif. Cette énergie régit la distance entre les différents points de la courbe. Nous avons fait varier la valeur du α (α= {0.1, 0.5, 1.0}) et fixé les valeurs des autres paramètres (β = 0.45, ɣ =0.5).



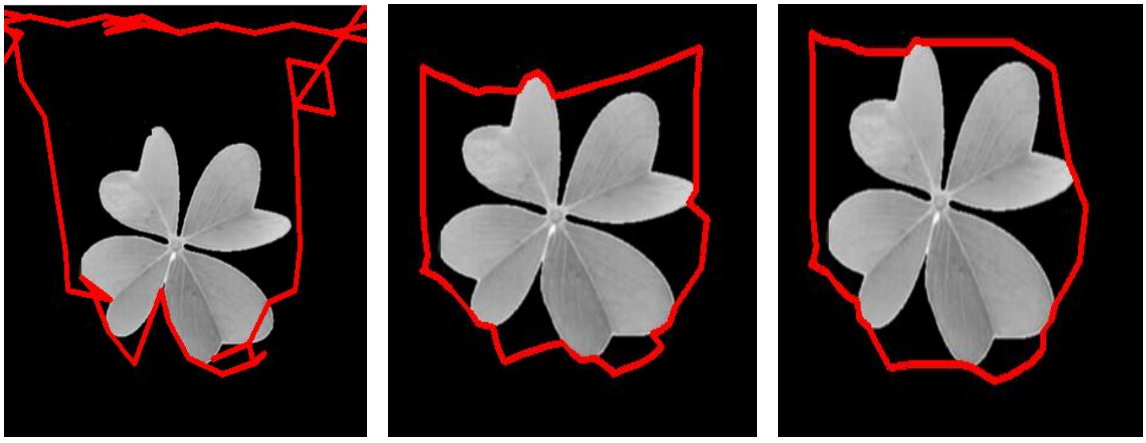
α = 0.1 α = 0.5 α = 1.0

Figure 3.11: Influence de α sur le résultat de la segmentation par cvSnake.

Nous remarquons que lorsque (α = 1.0), le contour final a détecté la position de l'objet ; mais le résultat reste non concluant. L'évolution de la courbe dépend de la valeur d'élasticité. On constate que la courbe ne converge pas à l'intérieure des concavités.

B. Tests sur le paramètre de courbure Beta :

Rappelons que β est le paramètre qui permet de contrôler l'énergie de courbure du snake. Nous avons fixé les valeurs des paramètres suivants α =0.35 et ɣ = 0.5 ; et nous avons varié la valeur du coefficient β = {0, 0.5, 1.0}.



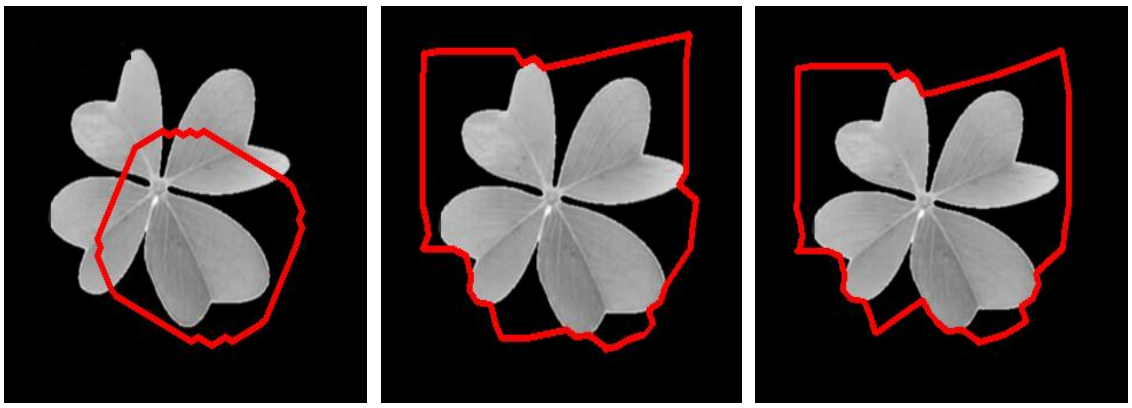
β = 0 β = 0.5 β = 1.0

Figure 3.12: Influence de β sur le résultat de la segmentation par cvSnake.

D'après les résultats obtenus dans la figure III.12, nous remarquons que lorsque (β = 1.0), le contour final de cvSnake situe sur les frontières de l'objet sans donner la forme exacte ; mais lorsque β = 0, la courbe se déforme et n'arrive pas à extraire la forme de l'objet.

C. Tests sur le paramètre de gradient Gamma :

Le Gradient est la deuxième énergie du contour actif qui dépend de l'image (externe). ɣ est le coefficient qui contrôle cette énergie. Là nous avons varié cette valeur (ɣ = {0, 0.5, 1.5}) et fixé les valeurs de α et β (α =0.35, β =0.45).



ɣ = 0 ɣ = 0.5 ɣ = 1.0

Figure 3.13: Influence de ɣ sur le résultat de segmentation par cvSnake.

La comparaison entre les résultats obtenus dans la figure précédente montre une importance du paramètre ɣ. Si (ɣ = 0), la courbe ne donne aucune résultat (ne connaissais pas la forme de l'objet). Par contre quand (ɣ > 0), la courbe commence à situer sur les frontières de l'objet d'intérêt.

3.4.1.2 Tests sur l'influence des paramètres de Greedy :

Pour montrer l'importance des paramètres de l'algorithme de Greedy : (α, β, ɣ & δ), nous avons fait une série de tests sur une image présentant des concavités en faisant varier les valeurs de ses paramètres.

A. Tests sur le paramètre de continuité Alpha :

Nous avons fait varier la valeur du coefficient α (α= {0.1, 0.5, 1.0}) et fixé les valeurs des autres paramètres (β = 1.2, ɣ =1.5, δ =3.0).

α = 0.1 α = 0.5 α = 1.0

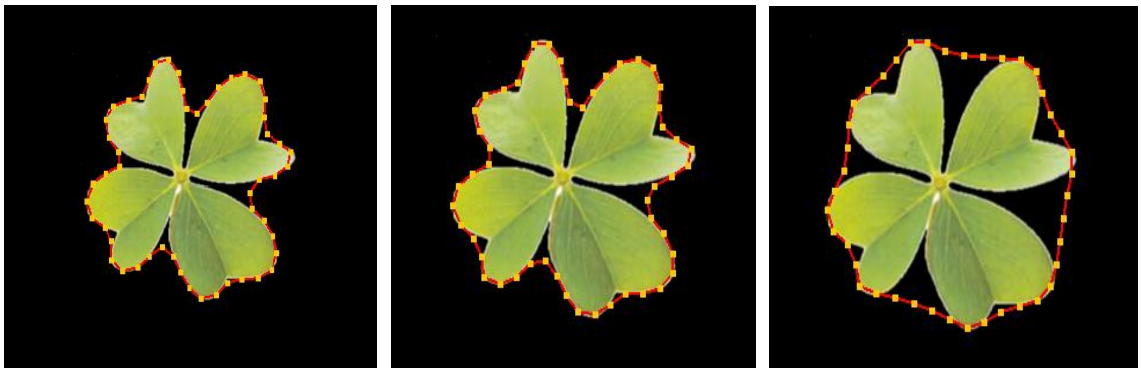
Figure 3.14: Influence du α sur le résultat de la segmentation par Greedy.

D'après la figure III.14, nous constatons que les résultats obtenus sont presque les mêmes pour les différentes valeurs de α. Cela est dû à l'opération de réchantillonnage effectuée à la fin de chaque itération de l'algorithme de Greedy.

Rappelons que l'opération de réchantillonnage consiste à créer ou supprimer des points de façon à conserver une distance homogène entre les sommets voisins.

B. Tests sur le paramètre de courbure Beta :

Nous avons fixé les valeurs des paramètres suivants α =1.0, ɣ = 1.5 et δ= 3.0 et nous faisons varier la valeur du coefficient β = {0, 0.5, 2.0}.



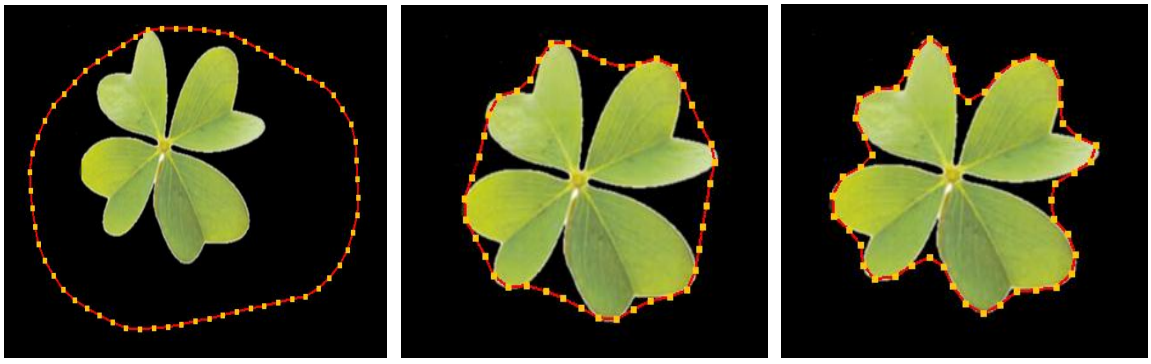
β = 0 β = 0.5 β = 2.0

Figure 3.15: Influence de β sur le résultat de la segmentation par Greedy.

La comparaison des résultats de segmentations obtenus sur la figure ci-dessus montre que la qualité du contour détecté est proportionnelle à β. Plus on diminue la valeur de β plus la courbure du contour actif augmente. Nous remarquons qu'avec une valeur de β égale 0 le contour a détecté la forme exacte de l'objet, au contraire quand la valeur de β augmente la courbe va tendre vers un cercle.

C. Tests sur le paramètre de gradient Gamma :

Nous varions maintenant γ (Ɣ = {0, 0.5, 1}) et nous fixons : (α=1.0, β=1.0, δ=3.0).



γ = 0 γ = 0.5 γ = 1.5

Figure 3.16: Influence de γ sur le résultat de segmentation par Greedy.

La comparaison visuelle des résultats obtenus, (figure III.16**)** montre l'importance de γ. En effet dans le cas où nous négligeons le gradient (γ =0), nous remarquons que la courbe ne capte aucun contour. Ce test montre bien que sans gradient, le contour actif ne peut pas détecter les points de contours d'objets. Par contre avec (γ > 0), la courbe

commence à reconnaitre la forme de l'objet. Cependant avec une grande valeur de ɣ, une faible variation de niveau de gris (faible bruit) peut stopper la progression de la courbe.

D. Tests sur le paramètre de ballon Delta :

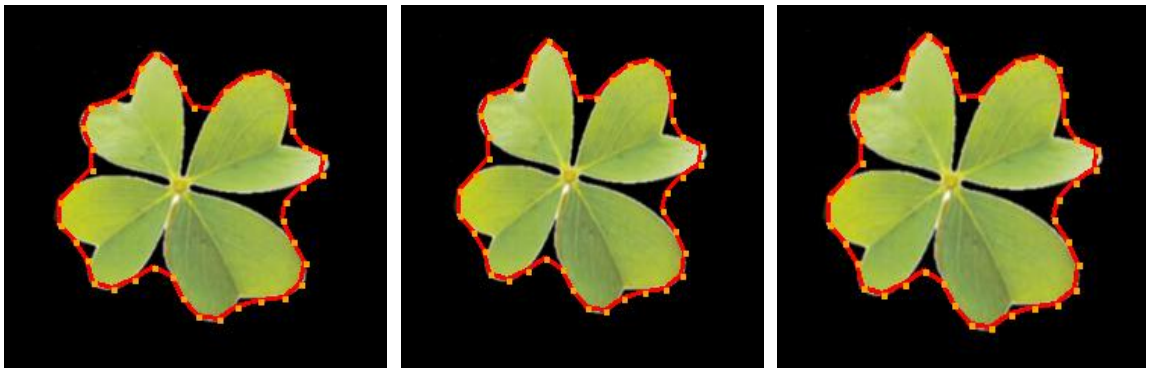
Cette énergie permet d'introduire des connaissances à priori sur ce qu'on cherche. Un coefficient d'énergie de ballon positif va gonfler le Snake, alors qu'un coefficient négatif va le rendre expansif.

Nous fixons : α, β et ɣ (α = 1.0, β = 1.2, ɣ = 1.5), et on varie δ de deux manières :

Des valeurs positives : pour δ > 0, la courbe va tendre vers le barycentre du cercle.

Des valeurs négatives : pour δ < 0, alors la courbe va gonfler si le Snake est initialisé à l'intérieur de l'objet.

- Des valeurs positives : (δ = {0, 3.0, 5.0})



δ = 0 δ = 3.0 δ = 5.0

- Des valeurs négatives : (δ = {-5.0, -3.0, -1})

δ = -1.0 δ = -3.0 δ = -5.0

Figure 3.17: Influence de δ sur le résultat de segmentation par Greedy.

D'après les résultats obtenus (figure 3.17), nous constatons que le paramètre δ n'a pas de grande influence sur le type d'image que nous utilisons. En effet dans le cas où nous négligeons l'énergie de ballon (δ =0), nous remarquons que la courbe ne subit qu'un petit changement.

3.4.2 Tests sur une image synthétisée sans présence de bruit :

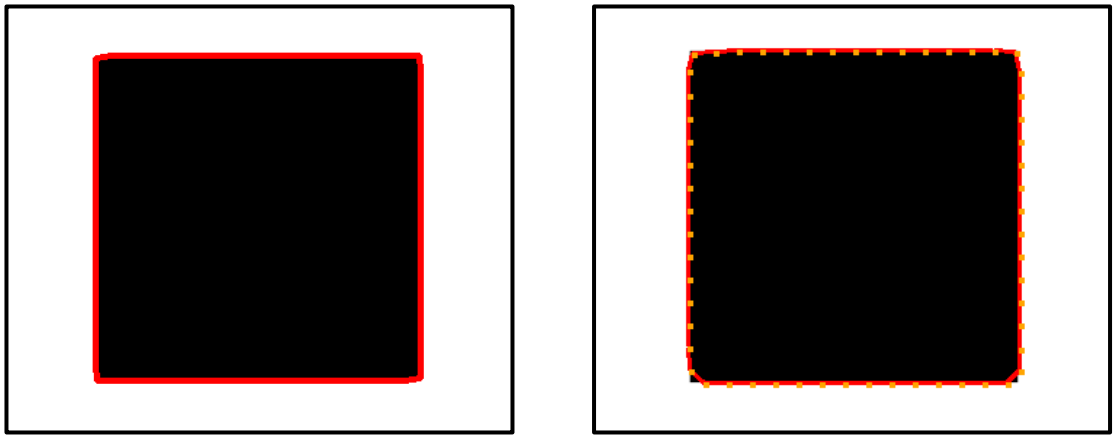
Dans ce qui suit nous testons des images de synthèse avec un objet simple sur un fond uniforme non texturé (un carré noir sur un fond blanc).

Pour ces tests nous avons utilisé les paramètres suivants :

Paramètres de cvSnake	Paramètres de Greedy
α = 0.35, β = 0.45, γ = 0.5 Nbr_points = 100 Max_Itérat = 500	α = -1.0, β = 1.0, γ = 1.5, δ = 3.0 Dmax = 16, Dmin = 8, Max_itérat = 300 Seuil = 15

Tableau 3.1 : Paramètres utilisés pour l'image de synthèse simple sans bruit.

La figure 3.18 illustre la détection de contours en appliquant les deux méthodes cvSnake et Greedy.



(a) (b)

Figure 3.18: Détection de contour d'un objet simple sans bruit par les méthodes : a) cvSnake, b) Greedy.

Nous avons obtenu des résultats satisfaisants pour les deux méthodes. La qualité de segmentation obtenue par ces deux méthodes est presque équivalente. On remarque juste une petite différence dans les coins.

3.4.3 Tests sur une image synthétisée avec présence de bruit :

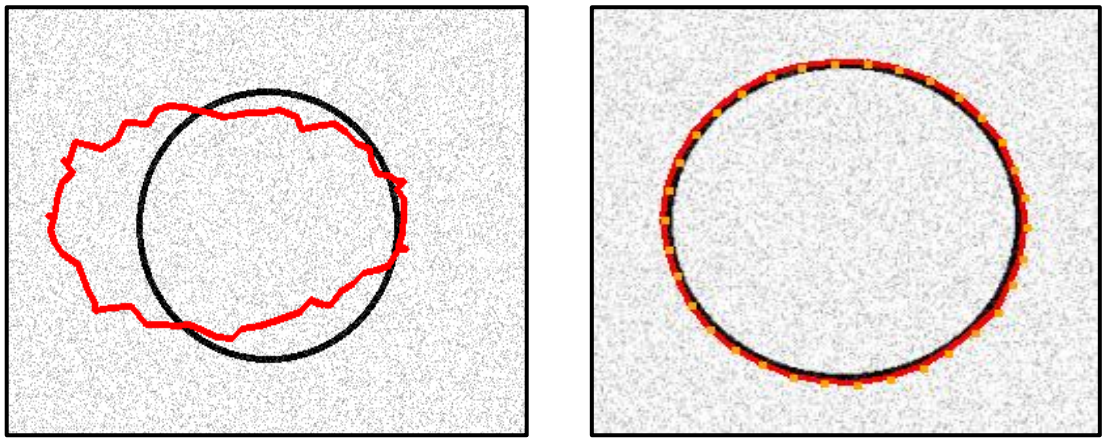
Dans ces tests, nous prenons des images de synthèse avec objet simple sur un fond texturé (présence de bruit).

Pour ces tests nous avons utilisé les paramètres suivants :

Paramètres de cvSnake	Paramètres de Greedy
α = 1.0, β = 1.2, γ =1.0 Nbr_points = 100 Max_Itérat = 500	α = 1.0, β = 1.2, γ = 1.5, δ = 3.0 Dmax = 16, Dmin =8, Max_itérat=300 Seuil = 30

Tableau 3.2 : Paramètres utilisés pour l'image de synthèse simple avec bruit.

La figure 3.19 illustre la détection de contours en appliquant les méthodes cvSnake et Greedy sur des images synthétisées bruitées.



(a) (b)

Figure 3.19 : Détection de contour d'un objet simple avec bruit par les méthodes : a) cvSnake, b) Greedy.

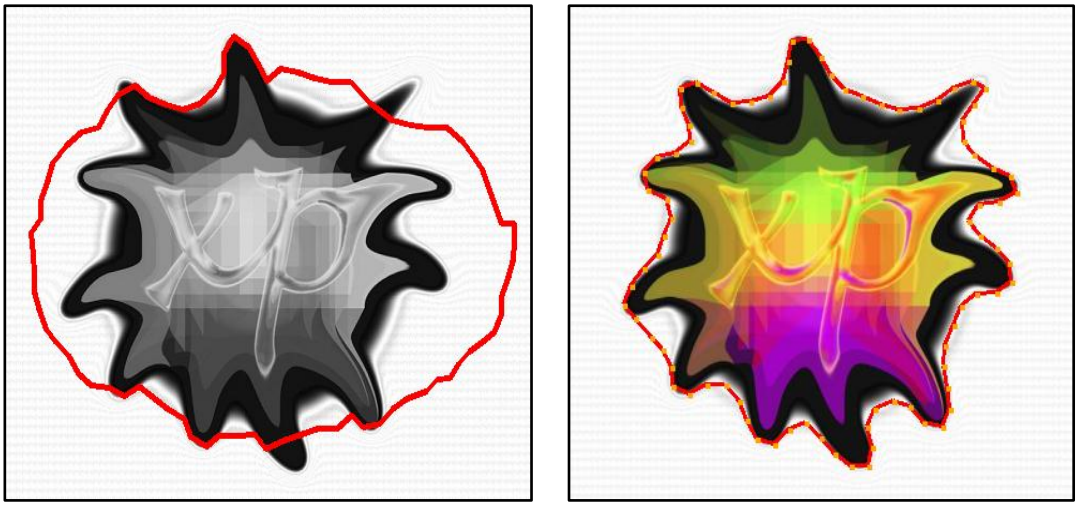
Nous constatons que le contour final de cvSnake n'épouse pas la forme du contour de l'objet. Le problème se situe au changement de NDG. Par contre le contour final de Greedy contient la forme exacte de l'objet. Dans ce cas d'image, les points de la courbe ne peuvent s'aventurer sans pénaliser leur énergie interne en augmentant leur courbure.

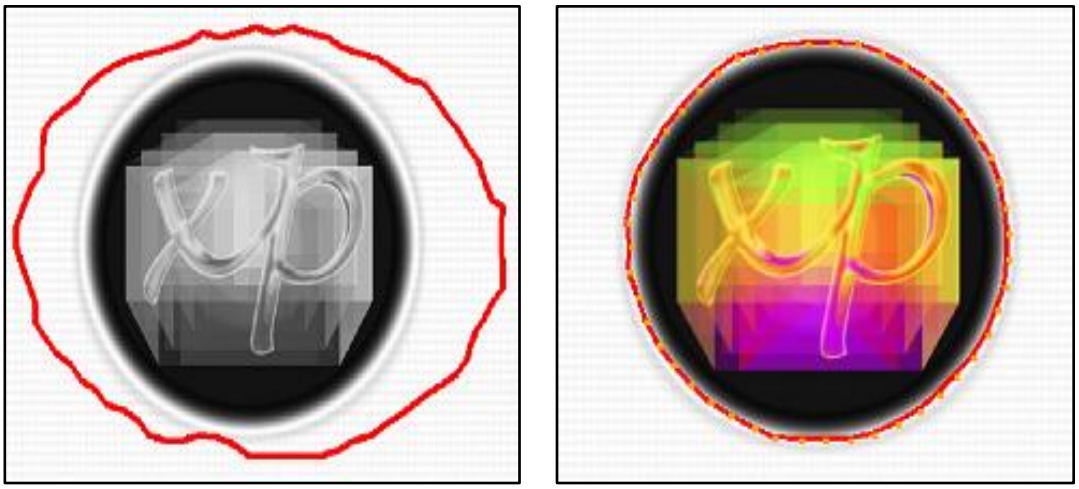
3.4.4 Tests sur des images avec concavités :

Les contours actifs ne donnent pas vraiment des bons résultats si l'objet à détecter présente des grandes concavités. Nous montrons ça à travers des tests que nous présentons ci-dessous ; nous utilisons pour ces tests une image présentant des concavités de différentes formes. Nous allons utiliser les paramètres suivants :

Paramètres de cvSnake	Paramètres de Greedy
α = 0.35, β = 0.45, γ =1.0 Nbr_points = 80 Max_Itérat = 500	α = -1.0, β = 0, γ = 1.5, δ =3.0 Dmax = 16, Dmin =8, Max_itérat=300 Seuil = 15

Tableau 3.3: Paramètres utilisés pour les images avec concavités.





(a) (b)

Figure 3.20 : Détection des contours des objets contient des concavités par les méthodes : a) cvSnake, b) Greedy.

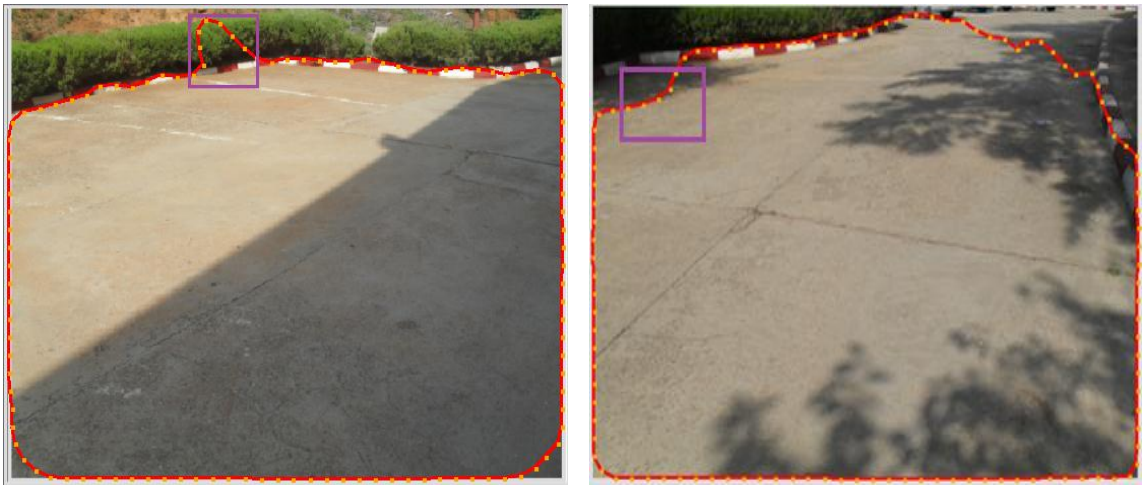
Conclusion :

D'après les tests que nous avons faits, nous avons montré que la méthode utilisée (cvSnake) ne donne pas des résultats concluants si l'image à segmenter présente des changements de NDG (texturée ou bruitée). C'est la raison pour laquelle nous avons utilisé la méthode de Greedy pour essayer d'améliorer les résultats de la segmentation et l'appliqué par la suite dans notre travail.

3.4.5 Tests sur des images d'extérieur pour l'extraction des chemins :

Dans ce qui suit nous testons l'algorithme de Greedy sur notre type d'image à savoir, les scènes d'extérieure montrant le chemin où peut naviguer notre robot mobile.

3.4.5.1 Tests sur des images couleurs :



(a) (b)

(c) (d)

Figure 3.21 : Détection des zones navigables dans un environnement structuré sur des images couleur.

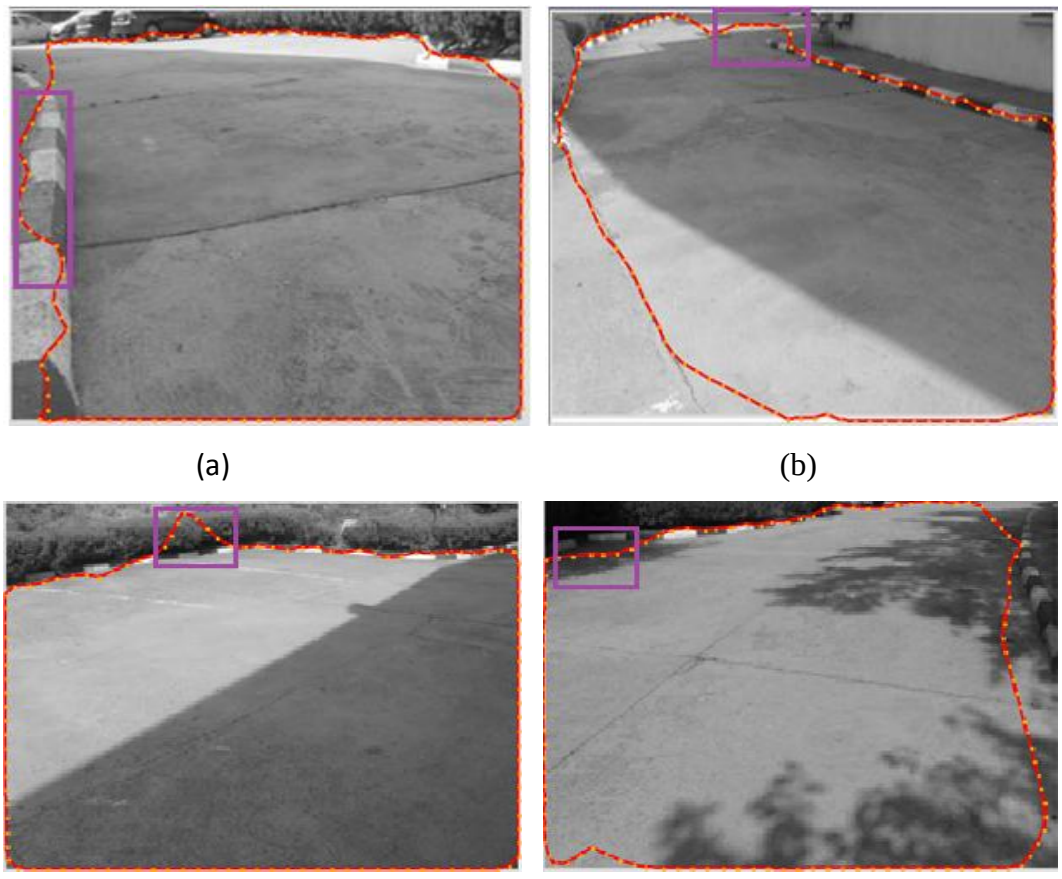
Pour les résultats obtenus, nous avons choisi des valeurs des différents paramètres expérimentalement pour avoir un meilleur résultat.

Le tableau qui suit contient toutes les valeurs des paramètres utilisées pour obtenir les résultats illustrés sur les images ci-dessus:

Images de tests	Alpha (courbure)	Beta (continuité)	Gamma (gradient)	Delta (ballon)	Max itératio	Chaque X itérations
(a)	-0.2	0	1.5	1.0	100	1
(b)	-1.0	0	10.5	3.0	200	2
(c)	-0.1	0	5.5	10.5	1000	1
(d)	-5.0	0	10.5	-3.0	300	1

Tableau 3.4: Paramètres utilisés pour les différentes images couleur.

3.4.5.2 Tests sur des images en NDG:



(a) (b)

(c) (d)

FIGURE 3.22: Détection des zones navigables dans un environnement structuré sur des images en NDG.

Les valeurs des paramètres ont été choisies expérimentalement pour garantir les bons résultats. Les paramètres utilisés pour ce test sont dans le tableau suivant :

Images de tests	Alpha (courbure)	Beta (continuité)	Gamma (gradient)	Delta (ballon)	Max itérati	Chaque X itérations
(a)	-0.3	0	10.5	-3.5	800	1
(b)	-0.3	0	10.5	-1.5	600	1
(c)	-10.5	0	10.5	12.0	200	1
(d)	-1.0	0	10.5	3.0	200	2

Tableau 3.5: Paramètres utilisés pour les différentes images en NDG.

3.4.6 Erreurs récurrentes lors de l'interprétation des d'images :

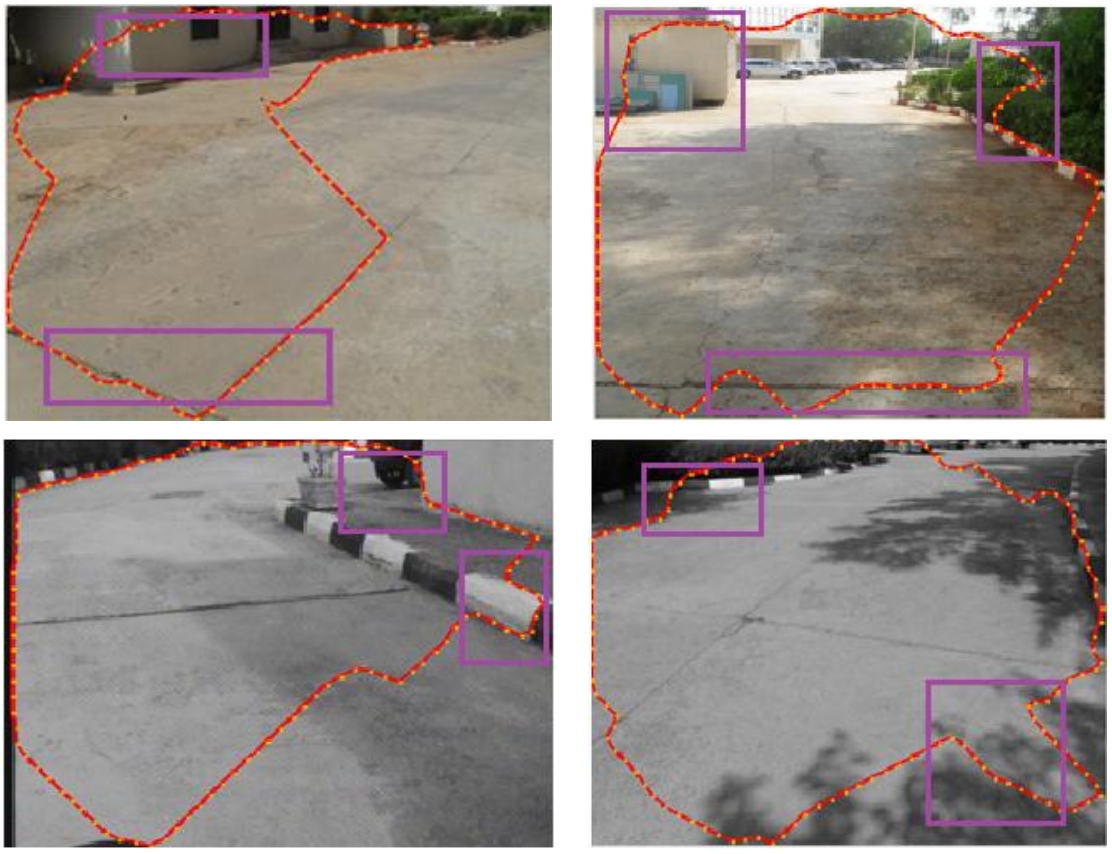


Figure 3.23 : Images contiennent des erreurs de segmentation et classification des régions lointaines, perturbation des contours par ombrages ou fausse trajectoire.

Les erreurs de segmentation sont montrées dans les figures précédentes par des rectangles. Parfois, il est très difficile d'avoir la bonne combinaison de paramètres garantissant un bon résultat, ceci est dû :

- Les nombres des paramètres à régler (α, β, ɣ et δ, le nombre d'itération, etc).
- Le choix de seuil pour la binarisation de l'image.
- L'initialisation automatique de la courbe dans l'image.

3.5 Conclusion :

Dans ce chapitre, nous avons présenté l'application développée, testé les deux méthodes implémentées cvSnake et Greedy à travers plusieurs images (simples, complexes et bruitées). On conclue que la qualité de l'image est un facteur important pour l'obtention de bon résultat de segmentation, ainsi que l'initialisation et le choix des paramètres.

Nous avons constaté que la méthode de Greedy a un inconvénient majeur, son pauvre convergence vers les régions à concavités. Nous avons montré aussi la sensibilité de ces deux méthodes au bruit.

Le problème majeur que nous avons rencontré est le choix des paramètres qui s'adaptent avec la région à segmenter ; c.à.d. la convergence du contour initial vers le contour de l'objet n'est possible que pour des paramètres α, β, γ et δ bien spécifiques (on ne peut pas choisir n'importe quelles valeurs).

Nous avons trouvé quelques inconvénients qui ont posé la plupart des difficultés rencontrées durant notre implémentation. La détection d'un chemin se réalise sans complications tant que les deux bordures (réelles) de la voie sont bien saillantes (bien contrastées). Ainsi, le chemin visualisé par le robot peut être incomplet (champ de vue limité), par exemple la situation où les limites de l'image sont considérées comme les contours d'un chemin qui est de plus en plus large. Dans cette situation, le robot se déplace au milieu du chemin ou la zone détectée dans l'image acquise. Et ainsi de suite jusqu'il arrive à sa destination finale.